

**VISVESVARAYA TECHNOLOGICAL  
UNIVERSITY**

**“JnanaSangama”, Belgaum -590014, Karnataka.**



**LAB RECORD**

**Computer Networks Laboratory (23CS5PCCON)**

*Submitted by*

**Likhith M (1BM22CS135)**

*in partial fulfillment for the award of the degree of*

**BACHELOR OF ENGINEERING**

*in*

**COMPUTER SCIENCE AND ENGINEERING**



**B.M.S. COLLEGE OF ENGINEERING**

**(Autonomous Institution under VTU)**

**BENGALURU - 560019**

**Academic Year 2024 - 25 (odd)**

# B.M.S. College of Engineering

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

## Department of Computer Science and Engineering



### CERTIFICATE

This is to certify that the Lab work entitled “Computer Networks (23CS5PCCON)” carried out by **Likhith M (1BM22CS135)**, who is bonafide student of **B.M.S. College of Engineering**. It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Laboratory report has been approved as it satisfies the academic requirements of the above-mentioned subject and the work prescribed for the said degree.

Srushti C S  
Assistant Professor  
Department of CSE, BMSCE

Dr. Kavitha Sooda  
Professor & HOD  
Department of CSE, BMSCE

# INDEX

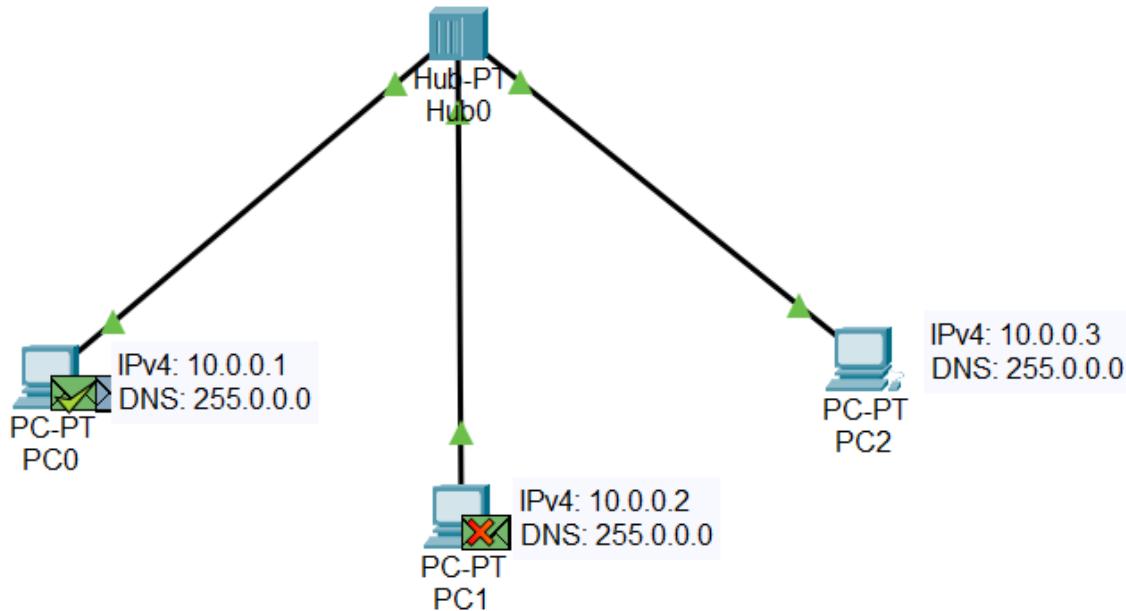
<b>Sl. No.</b>	<b>Date</b>	<b>Experiment Title</b>	<b>Page No.</b>
1	01.10.24	Laboratory Program – 1 (Topology Simulation)	1
2	08.10.24	Laboratory Program – 2 (Router IP Configuration)	5
3	08.10.24	Laboratory Program – 3 (Routing Configuration)	8
4	22.10.24	Laboratory Program – 4 (Default and Static Configuration)	11
5	29.10.24	Laboratory Program – 5 (TELNET Access)	16
6	29.10.24	Laboratory Program – 6 (TTL Demonstration)	18
7	12.11.24	Laboratory Program – 7(A) (DHCP Configuration within the same LAN)	22
8	12.11.24	Laboratory Program – 7(B) (DHCP Configuration outside the LAN)	25
9	12.11.24	Laboratory Program – 8 (Web Server & DNS)	28
10	19.11.24	Laboratory Program – 9 (RIP Routing Setup)	30
11	26.11.24	Laboratory Program – 10 (WLAN Setup)	33
12	26.11.24	Laboratory Program – 11 (ARP in LAN)	36
13	3.12.24	Laboratory Program – 12 (VLAN Configuration)	40
13	3.12.24	Laboratory Program – 13 (CRC Error Detection)	44

14	17.12.24	Laboratory Program – 14 (Leaky Bucket Algorithm)	48
15	24.12.24	Laboratory Program – 15(A) (TCP Client-Server)	50
16	24.12.24	Laboratory Program – 15(B) (UDP Client-Server)	52

Github Link: <https://github.com/01-BLUELOTUS/CN-1BM22CS135.git>

## LABORATORY PROGRAM – 1

Create a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting devices and demonstrate ping message.



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit (edit)	Delete
<input checked="" type="radio"/>	Successful	PC0	PC2	ICMP	<span style="background-color: #008000; border: 1px solid black; padding: 2px 5px;"> </span>	0.000	N	0		

```
c:\>ping 10.0.0.3

Pinging 10.0.0.3 with 32 bytes of data:

Reply from 10.0.0.3: bytes=32 time=9ms TTL=128
Reply from 10.0.0.3: bytes=32 time<1ms TTL=128
Reply from 10.0.0.3: bytes=32 time=1ms TTL=128
Reply from 10.0.0.3: bytes=32 time<1ms TTL=128

Ping statistics for 10.0.0.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 9ms, Average = 2ms
```

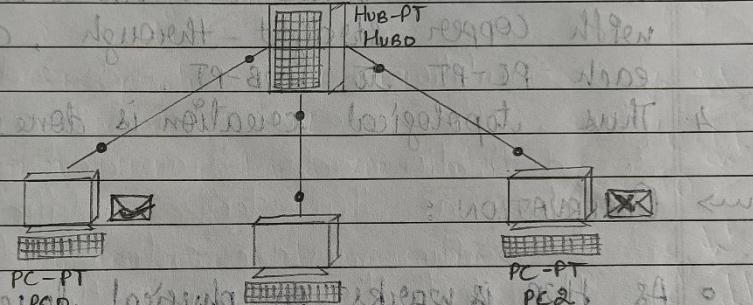
1.10.24

16.07.2023  
LABORATORY PROGRAM - 1

→ AIM OF THE EXPERIMENT : To demonstrate the transmission of a simple PDU between two devices connected using a hub and a switch.

→ TOPOLOGY :

◦ STAR TOPOLOGY [HUB] :



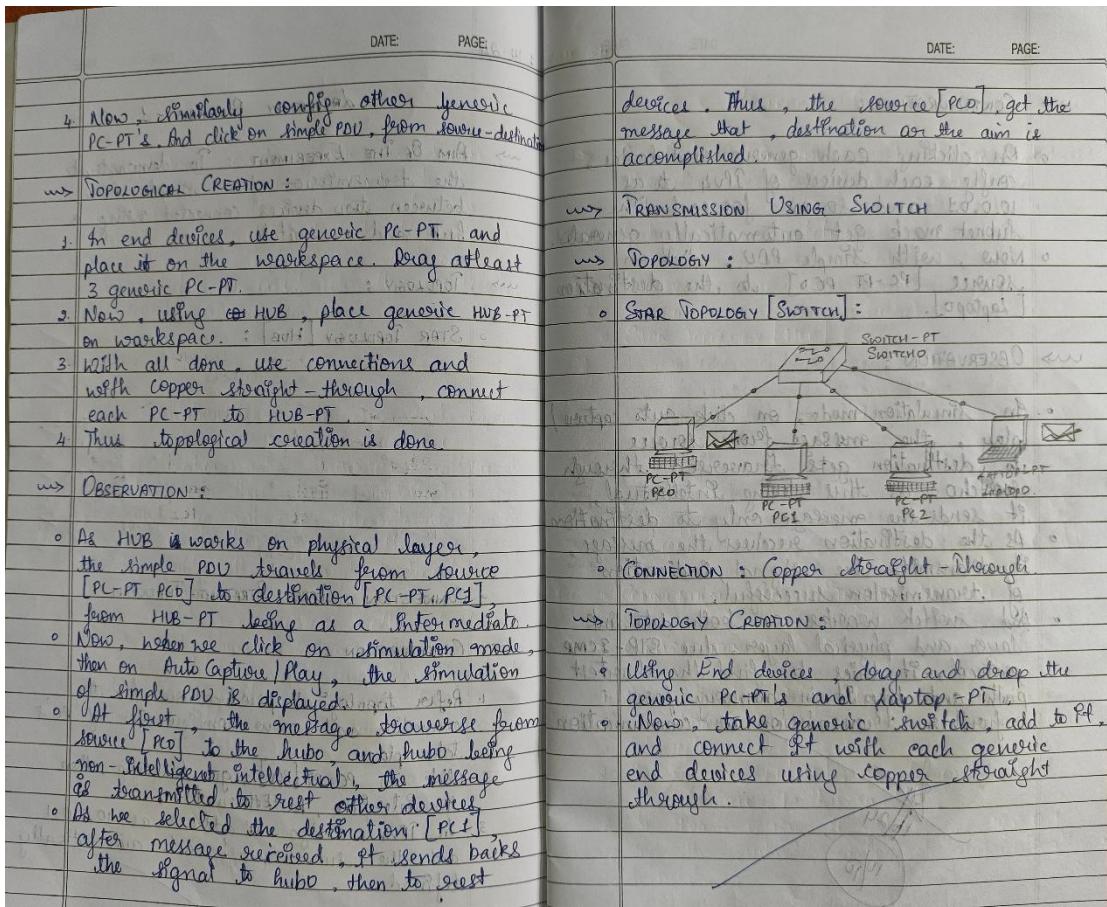
◦ CONNECTION : COPPER STRAIGHT-THROUGH

→ CONFIGURATION : a) click on connection config b)

1. Refer topological procedure for the connection and now configuration of PC's takes place.

2. Click on PC-PT [PCO], go to config, then the interface, fastethernet[1].

3. In IPv4, assign IP address as 10.0.0.1 and the subnet mask gets automatically defined.



→ CONFIGURATION :

- On clicking each generic end device, config each device of IPv4 to as 10.0.0.1 and so on for each. The subnet mask get automatically generated.
- Now, with simple PDU, click on source [PC-PT P00] to the destination [Laptop].

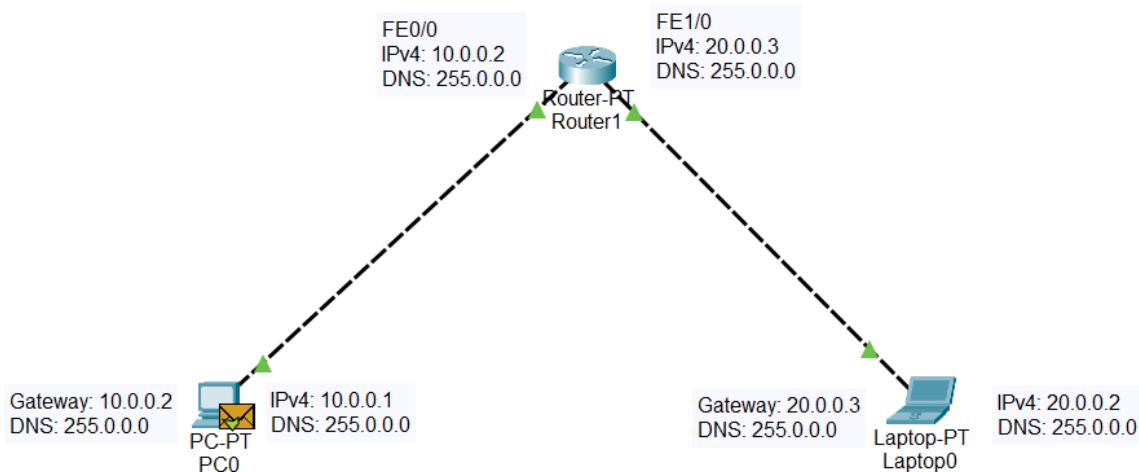
→ OBSERVATION :

- In simulation mode, on click auto capture / play, the message from source to destination gets traversed through switch, as this being Intelligent, it sends the message only to destination.
- As the destination receives the message, it transmits the signal indication of transmission successful.
- And switch works on both data link layer and physical layer, due STP - ICMP protocol, it tries to find the shortest path and keep it in memory for further transmission or communication.

10/24  
10/10

## LABORATORY PROGRAM – 2

Configure IP address to routers in packet tracer. Explore the following messages: ping responses, destination unreachable, request timed out, reply.



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC0	Laptop0	ICMP	■	0.000	N	0	(edit)	
●	In Progress	PC0	Laptop0	ICMP	■	0.000	N	1	(edit)	
●	In Progress	PC0	Laptop0	ICMP	■	0.000	N	2	(edit)	

```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 20.0.0.3

Pinging 20.0.0.3 with 32 bytes of data:

Reply from 20.0.0.3: bytes=32 time<1ms TTL=255

Ping statistics for 20.0.0.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

## LABORATORY PROGRAM - (Lab 8, 10, 11)

→ Config Aim Of The Experiment: Config IP address to router in packet tracer. Explain the following messages: ping responses, destination unreachable, request timed out, reply.

→ Configuration of interface: 8.0.0.1

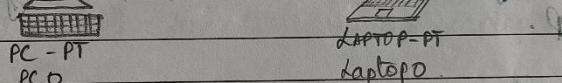
→ TOPOLOGY: 8.0.0.0 or 8.0.0.1 8.0.0.2

ROUTER-PT IP: 8.0.0.1  
ROUTER

bridge, 8.0.0.1 & 8.0.0.2 interface: SW1

via interface with both \* port at each

of the two ports, 8.0.0.1 & 8.0.0.2 interface



→ Connection: Copper Cross-Over: 8.0.0.1

8.0.0.2 8.0.0.0 or 8.0.0.1 8.0.0.2

→ TOPOLOGY CREATION: Neighboring one 8.0.0.1

→ Using end devices, drag and drop PC-PT and Laptop-PT.

→ Now, add Router-PT, and connect them using copper cross-over.

→ 8.0.0.0 or 8.0.0.1 8.0.0.2

→ 8.0.0.1, 8.0.0.2 8.0.0.0 or 8.0.0.1

8.0.0.0, 8.0.0.1 8.0.0.2

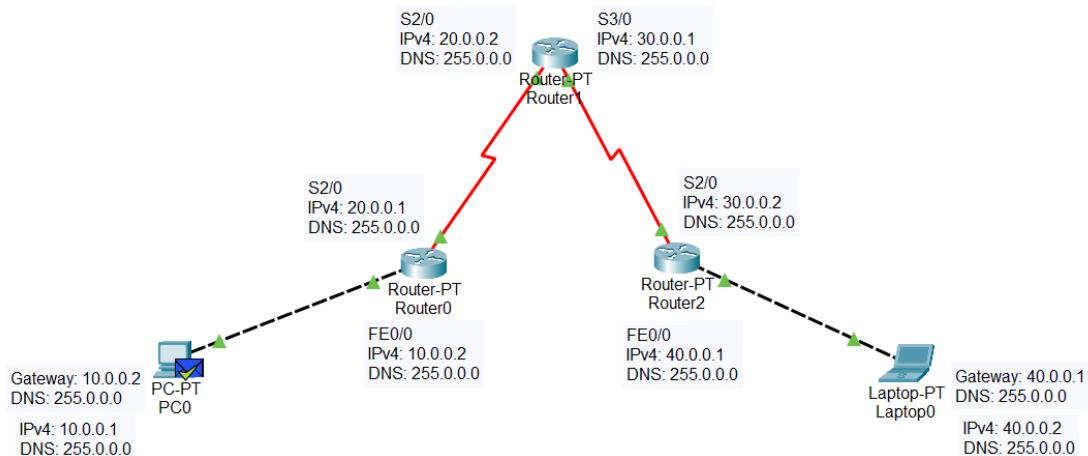
→ 8.0.0.1, 8.0.0.2 8.0.0.0 or 8.0.0.1

8.0.0.0, 8.0.0.1 8.0.0.2

DATE:	PAGE:
→ CONFIGURATION	
→ Router : Using CLI, config the left side interface [PC-A PC0]	type ping 20.0.0.2, at initial, out of 4 packets, 3 were received.
Step 1 : Enable	
Step 2 : config terminal	
Step 3 : Interface fastethernet 0/0	
Step 4 : IP address 10.0.0.2 255.0.0.0	
Step 5 : no shutdown	
Thus : Interface fastEthernet 0/0, changed state to up. And line protocol on interface fastEthernet 0/0, changed state to up.	
Similicely config the right side interface [laptop]	
Step 1 : interface fastethernet 1/0	
Step 2 : ip address 20.0.0.3 255.0.0.0	
Step 3 : no shutdown	
Thus, the protocol on interface fastEthernet 1/0, changed state to up.	
→ End Device : Configure PC-PT [PC0], IPv4 as 10.0.0.1 and gateway/DNS/IPv4 as 10.0.0.2.	
Now, config LAPTOP-PT [laptop], IPv4 as 20.0.0.2 and gateway/DNS IPv4 as 20.0.0.3.	
After configuration, lets send a message using ping.	
On clicking command prompt " in PC0,	
	BT

## LABORATORY PROGRAM – 3

Configure static route to the Router.



### SHOWIP ROUTE

```
C 10.0.0.0/8 is directly connected, FastEthernet0/0
C 20.0.0.0/8 is directly connected, Serial2/0
S 30.0.0.0/8 [1/0] via 20.0.0.2
S 40.0.0.0/8 [1/0] via 20.0.0.2
```

Figure 3.1: Router0

```
S 10.0.0.0/8 [1/0] via 20.0.0.1
C 20.0.0.0/8 is directly connected, Serial2/0
C 30.0.0.0/8 is directly connected, Serial3/0
S 40.0.0.0/8 [1/0] via 30.0.0.2
```

Figure 3.2: Router1

```
S 10.0.0.0/8 [1/0] via 30.0.0.1
S 20.0.0.0/8 [1/0] via 30.0.0.1
C 30.0.0.0/8 is directly connected, Serial2/0
C 40.0.0.0/8 is directly connected, FastEthernet0/0
```

Figure 3.3: Router3.3

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Laptop0	ICMP		0.000	N	0	(edit)	

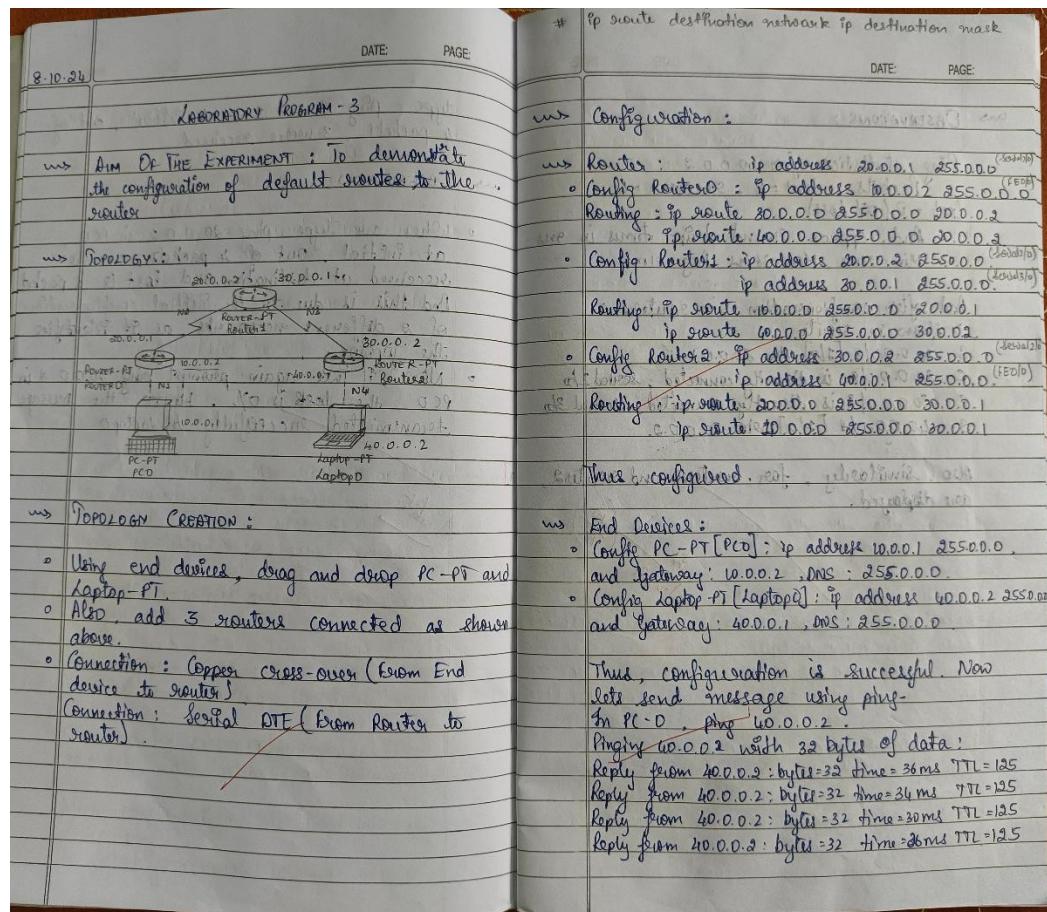
Cisco Packet Tracer PC Command Line 1.0  
C:\>ping 40.0.0.2

Pinging 40.0.0.2 with 32 bytes of data:

Reply from 40.0.0.2: bytes=32 time=36ms TTL=125  
Reply from 40.0.0.2: bytes=32 time=34ms TTL=125  
Reply from 40.0.0.2: bytes=32 time=30ms TTL=125  
Reply from 40.0.0.2: bytes=32 time=26ms TTL=125

Ping statistics for 40.0.0.2:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),  
Approximate round trip times in milli-seconds:  
Minimum = 26ms, Maximum = 36ms, Average = 31ms



## MS OBSERVATIONS :

Ping statistics for 192.0.0.2 :  
 Packets: Sent = 4, received = 4,  
 lost = 0 (0% loss).  
 And approx round trip times in ms.  
 Min = 20ms, Max = 36ms, Avg = 31ms.

And after configuration time reported is

> show ip route serial 2/0  
 S 0.0.0.0/8 [1/2] via 20.0.0.1 (eth0.1.1.1) serial 2/0  
 C 20.0.0.0/8 is directly connected, serial 2/0  
 C 20.0.0.0/8 is directly connected, serial 3/0  
 S 192.0.0.0/8 [1/0] via 192.0.0.2

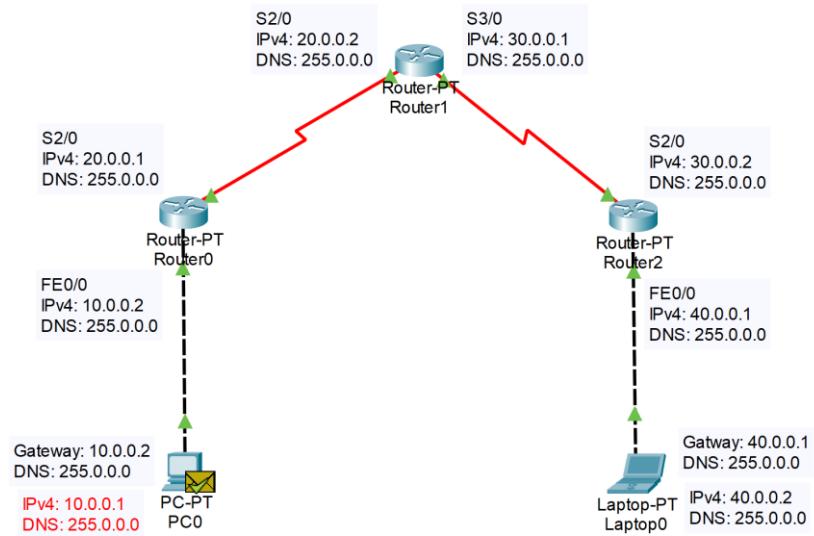
Also, similarly, for router and switch, are displayed.

~~192.0.0.0/8 [1/0] via 192.0.0.2 (eth0.1.1.1) serial 2/0~~  
~~192.0.0.0/8 [1/0] via 192.0.0.2 (eth0.1.1.1) serial 3/0~~  
~~192.0.0.0/8 [1/0] via 192.0.0.2 (eth0.1.1.1) serial 2/0~~  
~~192.0.0.0/8 [1/0] via 192.0.0.2 (eth0.1.1.1) serial 3/0~~

Q31. Infrastruktur in network layer, with  
 - when segment travel to  
 - so on only 0-18 m  
 - attach to link as when 192.0.0.0/8  
 P21 = STP 2nd ts = 2nd ts = e.g. 0.0.0.0 wait 100ms  
 2G1 = STP 2nd ts = 2nd ts = e.g. 0.0.0.0 wait 100ms  
 2B1 = STP 2nd ts = 2nd ts = e.g. 0.0.0.0 wait 100ms  
 2G1 = STP 2nd ts = 2nd ts = e.g. 0.0.0.0 wait 100ms

## LABORATORY PROGRAM – 4(A)

Configure default route, static route to the Router.



### SHOW IP ROUTE

Gateway of last resort is 20.0.0.2 to network 0.0.0.0

```

C      10.0.0.0/8 is directly connected, FastEthernet0/0
C      20.0.0.0/8 is directly connected, Serial2/0
S*     0.0.0.0/0 [1/0] via 20.0.0.2
  
```

*Figure 4.1: Router0*

```

S      10.0.0.0/8 [1/0] via 20.0.0.1
C      20.0.0.0/8 is directly connected, Serial2/0
C      30.0.0.0/8 is directly connected, Serial3/0
S      40.0.0.0/8 [1/0] via 30.0.0.2
  
```

*Figure 4.2: Router1*

Gateway of last resort is 30.0.0.1 to network 0.0.0.0

```

C      30.0.0.0/8 is directly connected, Serial2/0
C      40.0.0.0/8 is directly connected, FastEthernet0/0
S*     0.0.0.0/0 [1/0] via 30.0.0.1
  
```

*Figure 4.3: Router2*

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Laptop0	ICMP		0.000	N	0	(edit)	

```
C:\>ping 40.0.0.2
```

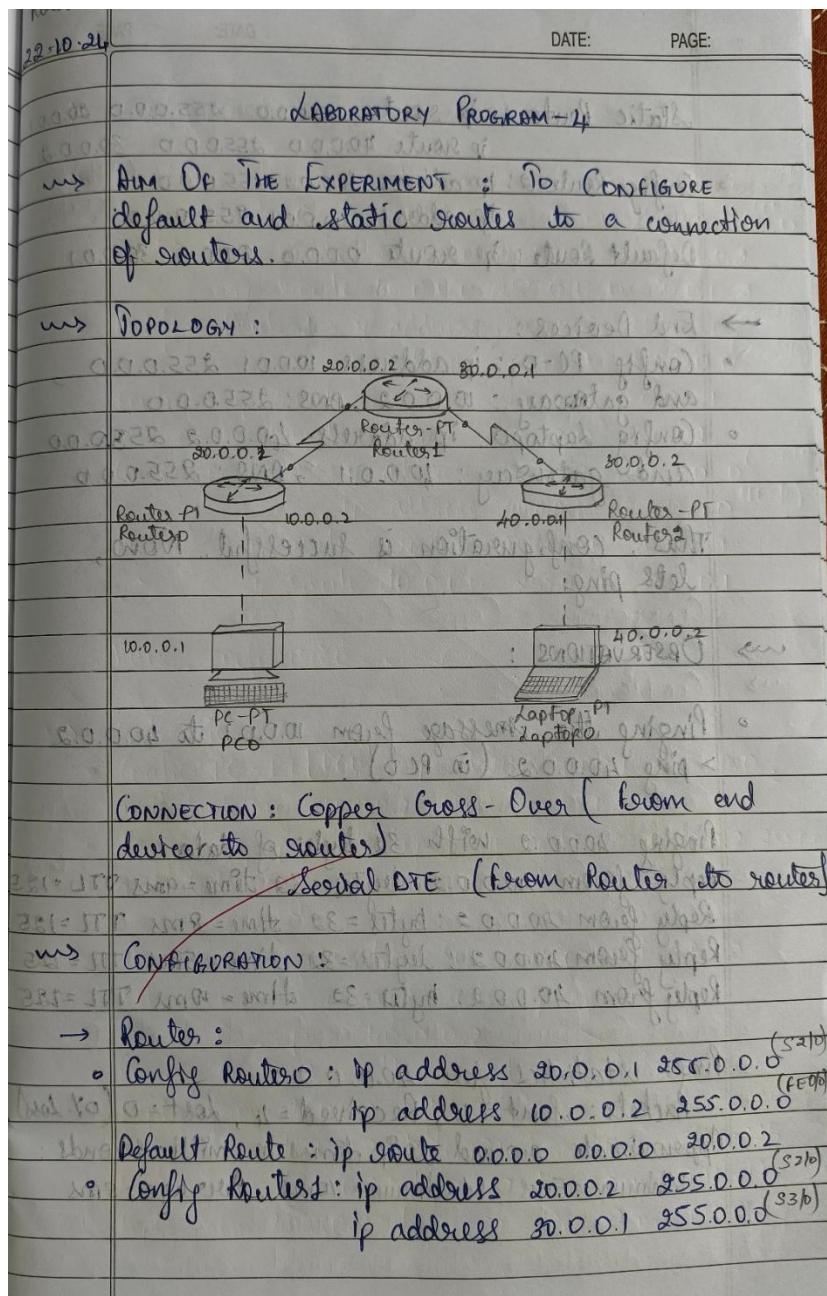
```
Pinging 40.0.0.2 with 32 bytes of data:
```

```
Reply from 40.0.0.2: bytes=32 time=34ms TTL=125  
Reply from 40.0.0.2: bytes=32 time=33ms TTL=125  
Reply from 40.0.0.2: bytes=32 time=30ms TTL=125  
Reply from 40.0.0.2: bytes=32 time=33ms TTL=125
```

```
Ping statistics for 40.0.0.2:
```

```
PACKETS: Sent = 4, Received = 4, Lost = 0 (0% loss),  
Approximate round trip times in milli-seconds:
```

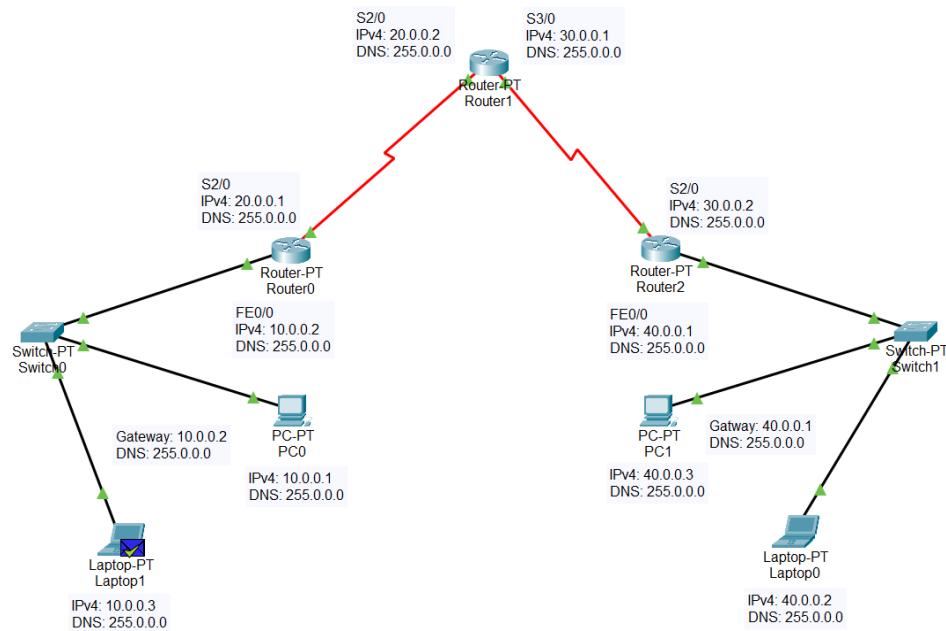
```
Minimum = 30ms, Maximum = 34ms, Average = 32ms
```



<p>DATE: PAGE: 11</p> <p>Static Routes: ip route 10.0.0.0 255.0.0.0 20.0.0.1 ip route 40.0.0.0 255.0.0.0 30.0.0.2</p> <ul style="list-style-type: none"> <li>Config Router 2: ip address 30.0.0.3 255.0.0.0 [2/0]</li> <li>ip address 40.0.0.1 255.0.0.0 [FE/0]</li> </ul> <p>Default Route: ip route 0.0.0.0 0.0.0.0 20.0.0.1</p> <p>→ End Devices:</p> <ul style="list-style-type: none"> <li>Config PC-0: ip address 10.0.0.1 255.0.0.0, and gateway: 20.0.0.2, DNS: 255.0.0.0</li> <li>Config Laptop: ip address 40.0.0.2 255.0.0.0, and gateway: 10.0.0.1, DNS: 255.0.0.0</li> </ul> <p>Thus, configuration is successful. Now, let's ping.</p> <p>→ OBSERVATIONS :</p> <ul style="list-style-type: none"> <li>Pinging the message from 10.0.0.1 to 40.0.0.2. &gt; ping 40.0.0.2 (in PC-0).</li> <li>Pinging 40.0.0.2 with 32 bytes of data: Reply from 40.0.0.2: bytes=32 time=4ms TTL=125 Reply from 40.0.0.2: bytes=32 time=8ms TTL=125 Reply from 40.0.0.2: bytes=32 time=5ms TTL=125 Reply from 40.0.0.2: bytes=32 time=10ms TTL=125</li> <li>Ping statistics for 40.0.0.2: packets: 4 received = 4, lost = 0 (0% loss). Approximate round trip times in milliseconds: Minimum=5ms, Maximum=10ms, Average=8ms</li> </ul>	<p>DATE: PAGE: 11 - 12</p> <p>And after configuration is in routers:</p> <p>&gt; show ip route</p> <p>C 10.0.0.0/8 is directly connected, FastEthernet0/0 C 20.0.0.0/8 is directly connected, Serial2/0 S* 0.0.0.0/0 [1/0] via 20.0.0.2</p> <p>And similarly, for routers and router 2 are displayed.</p> <p><i>(Handwritten notes in the right margin)</i></p>
--	---

## LABORATORY PROGRAM – 4(B)

Configure default route, static route to the Router, inclusive switches.



### SHOW IP ROUTE

```
Gateway of last resort is 20.0.0.2 to network 0.0.0.0
```

```
C      10.0.0.0/8 is directly connected, FastEthernet0/0
C      20.0.0.0/8 is directly connected, Serial2/0
S*     0.0.0.0/0 [1/0] via 20.0.0.2
```

*Figure 4.1: Router0*

```
S      10.0.0.0/8 [1/0] via 20.0.0.1
C      20.0.0.0/8 is directly connected, Serial2/0
C      30.0.0.0/8 is directly connected, Serial3/0
S      40.0.0.0/8 [1/0] via 30.0.0.2
```

*Figure 4.2: Router1*

```
Gateway of last resort is 30.0.0.1 to network 0.0.0.0
```

```
C      30.0.0.0/8 is directly connected, Serial2/0
C      40.0.0.0/8 is directly connected, FastEthernet0/0
S*     0.0.0.0/0 [1/0] via 30.0.0.1
```

*Figure 4.3: Router2*

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Laptop0	ICMP		0.000	N	0	(edit)	

```
C:\>ping 40.0.0.3

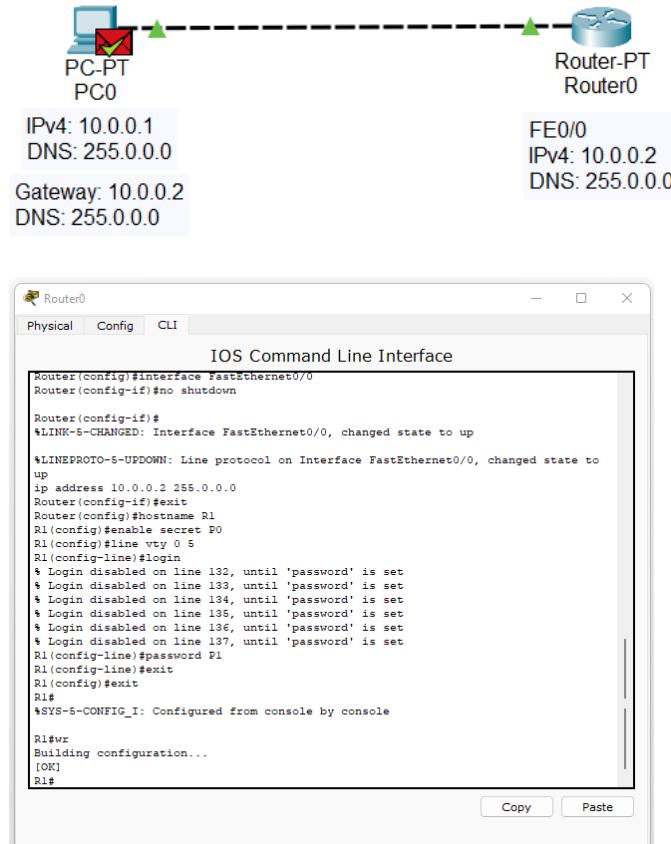
Pinging 40.0.0.3 with 32 bytes of data:

Reply from 40.0.0.3: bytes=32 time=35ms TTL=125
Reply from 40.0.0.3: bytes=32 time=37ms TTL=125
Reply from 40.0.0.3: bytes=32 time=24ms TTL=125
Reply from 40.0.0.3: bytes=32 time=38ms TTL=125

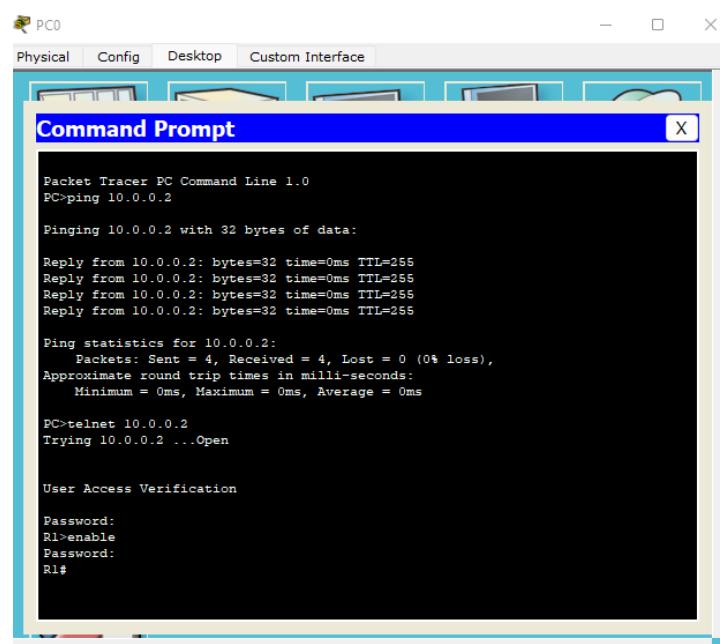
Ping statistics for 40.0.0.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 24ms, Maximum = 38ms, Average = 33ms
```

## LABORATORY PROGRAM – 5

To understand the operation of TELNET by accessing the router in server room from a PC in IT office.



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Router0	ICMP	■	0.000	N	0	(edit)	



<p style="text-align: right;">DATE: PAGE:</p> <p>29-10-24</p> <p><b>LABORATORY EXPERIMENT - 5/9</b></p> <p>→ <b>HIM OF THE EXPERIMENT :</b> To understand the operation of TELNET by accessing the Router to placed in the lab accessing secure screen from a PC in IT Office.</p> <p>→ <b>TOPOLOGY :</b></p> <p>CONNECTION : Copper Cross - Desc.</p> <p>→ <b>CONFIGURATION :</b></p> <ul style="list-style-type: none"> <li>• After topology creation.</li> <li>• Config PC0 and Router with corresponding address</li> <li>• Router config : ip address 10.0.0.2 255.0.0.0</li> <li>• PC0 config: ip address 10.0.0.1 255.0.0.0 And gateway as : 10.0.0.2 255.0.0.0.</li> <li>• And Configuring the config of Router0.</li> </ul> <pre> # hostname R1 # enable secret 50 # line vty 0 5 [virtual terminal is allowed for lines 0,1,2,3,4,5] # login </pre> <p>% Login disabled on line 132, until 'password' is set. % Login disabled on line 137, until 'password' is set.</p>	<p style="text-align: right;">DATE: PAGE:</p> <p># line vty 0 5 [virtual terminal is allowed for lines 0,1,2,3,4,5]</p> <p># login</p> <p># password P1</p> <p>R1# use [Building configuration]</p> <p>→ <b>TOPOLOGY CREATION :</b></p> <ul style="list-style-type: none"> <li>• Make the connection with PC and Router and copper - cross desc</li> </ul> <p>→ <b>OBSERVATION :</b></p> <ul style="list-style-type: none"> <li>• After configuration, pinging message from PC0 to R1</li> <li>&gt; ping 10.0.0.2. Pinging 10.0.0.2 with 32 bytes of data: Reply from 10.0.0.2: bytes = 32 time = 0ms TTL = 255 Reply from 10.0.0.2: bytes = 32 time = 0ms TTL = 255</li> </ul> <p>Ping statistics for 10.0.0.2: Packets: Sent = 4 Received = 4 Lost = 0 (0% loss), Approximate round trip times in milli-seconds: Min = 0ms, Max = 0ms, Avg = 0ms</p> <p>PC&gt; telnet 10.0.0.2 Trying 10.0.0.2... Open</p> <p>User Access Verification Password: P1 R1's enable Password: P0 R1#</p> <p>Now, we are able to access R1 and make configurations.</p>
--	--

# LABORATORY PROGRAM – 6

Demonstrate the TTL/ Life of a Packet.

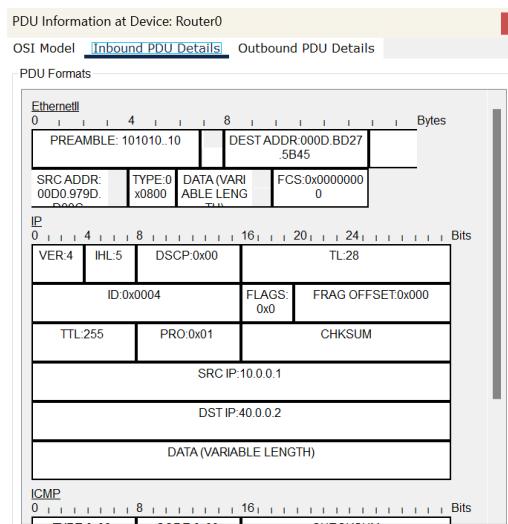
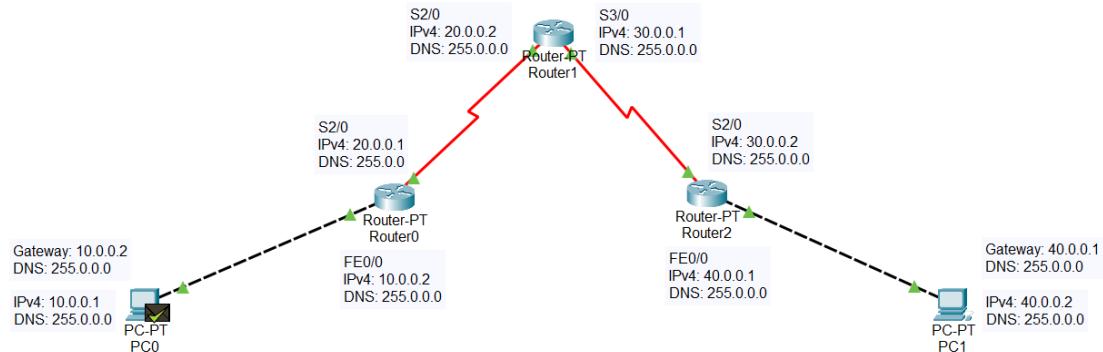


Figure 6.1: Inbound PDU, Router0

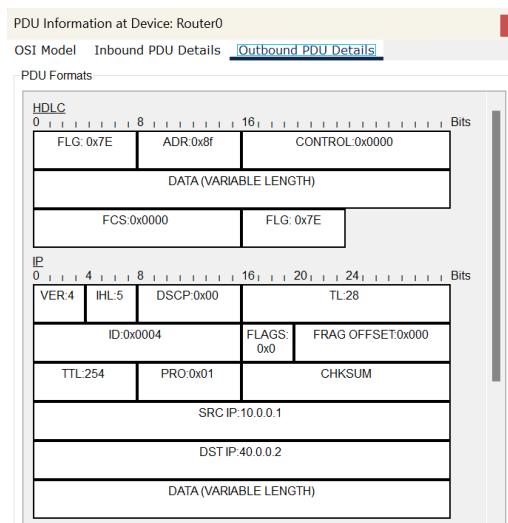


Figure 6.2: Outbound PDU, Router0

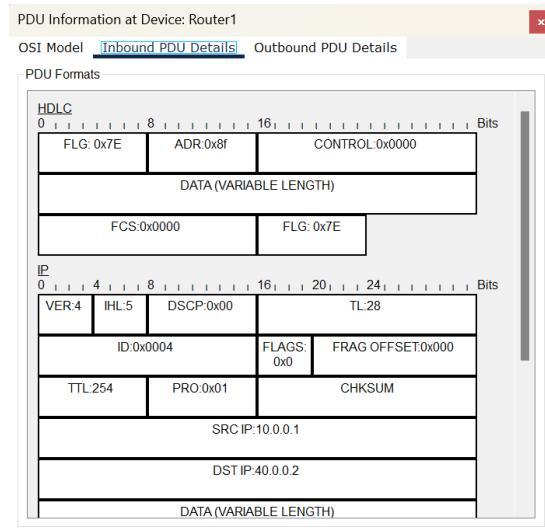


Figure 6.3: Inbound PDU, Router1

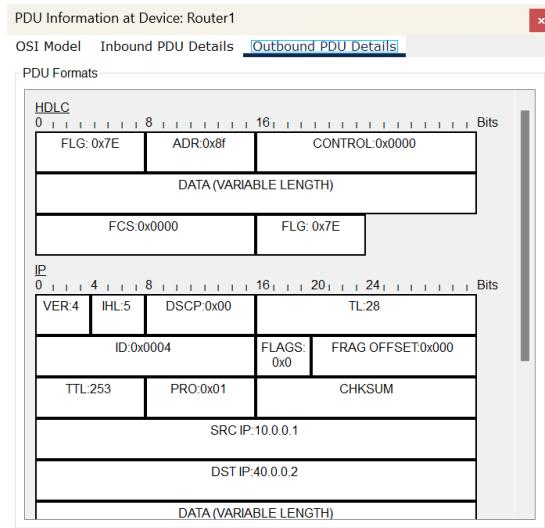


Figure 6.4: Outbound PDU, Router1

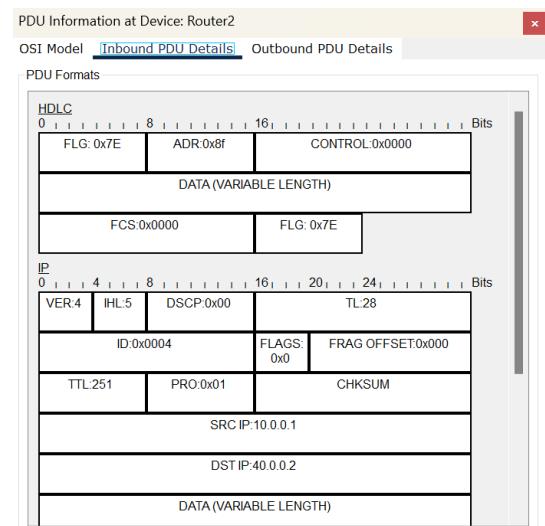


Figure 6.5: Inbound PDU, Router2

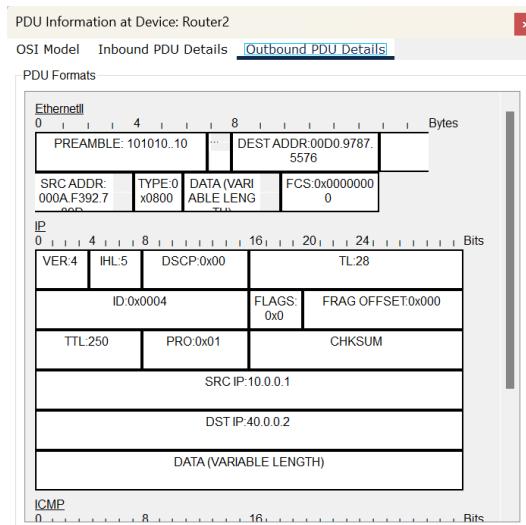


Figure 6.6: Outbound PDU, Router2

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	PC1	ICMP		0.000	N	0	(edit)	

```
c:\>ping 40.0.0.2

Pinging 40.0.0.2 with 32 bytes of data:

Reply from 40.0.0.2: bytes=32 time=72ms TTL=123
Reply from 40.0.0.2: bytes=32 time=53ms TTL=123
Reply from 40.0.0.2: bytes=32 time=55ms TTL=123
Reply from 40.0.0.2: bytes=32 time=69ms TTL=123

Ping statistics for 40.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 53ms, Maximum = 72ms, Average = 62ms
```

<p>29-10-26</p> <p>LABORATORY PROGRAM - 6</p> <p><del>AIM OF THE EXPERIMENT : Demonstrate the TTL / life of a packet</del></p> <p><del>TOPOLOGY :</del></p> <p><del>Connections : Copper Cross - Over (From end device to Router) + bus + serial serial, ATM (From Router to Router)</del></p> <p><del>Input - Out Lines PPTP, L2TP = ATM</del></p> <p><del>POPULATION :</del></p> <p><del>Configure a network as mentioned in diagram.</del></p> <p><del>CONFIGURATION :</del></p> <ul style="list-style-type: none"> <li>Config all routers and end devices accordingly as done in previous experiment with default and static routing.</li> </ul>	<p>DATE: PAGE: 41/81</p> <p>OBSERVATION:</p> <ul style="list-style-type: none"> <li>After configuration and pingng message from 10.0.0.1 to 10.0.0.3 using Simple Pov.</li> <li>In simulation mode, On clicking Pov at initial stage, at Inbound Pov details the TTL: 255 and after 1 hop, the Outbound Pov details on TTL: 254. Thus, at each hop, the TTL is decreased by one.</li> <li>At destination, in PC1, the TTL is 250, means there is a decrement of TTL at each hop.</li> <li>The message is deassembled successfully.</li> <li>And if the TTL is less than required hop, the packet is discarded, and again a new message has to be sent.</li> </ul>
--	---

## LABORATORY PROGRAM – 7(A)

To Configure IP addresses of the host using DHCP server within a LAN.

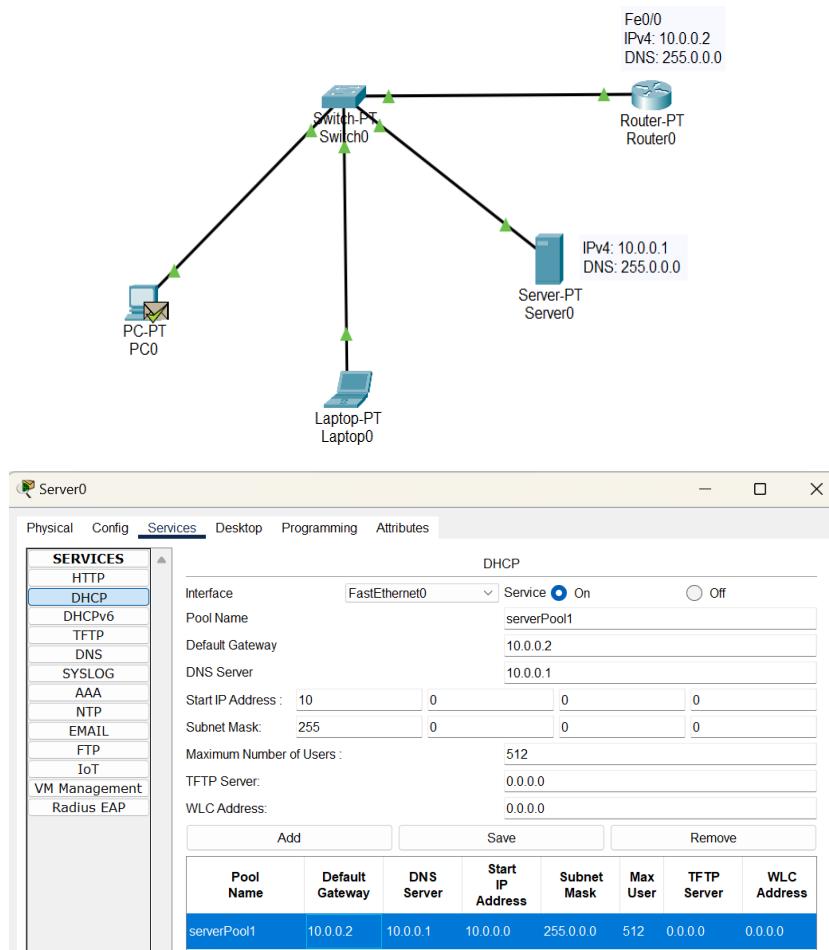


Figure 7.1: DHCP Service, Server0

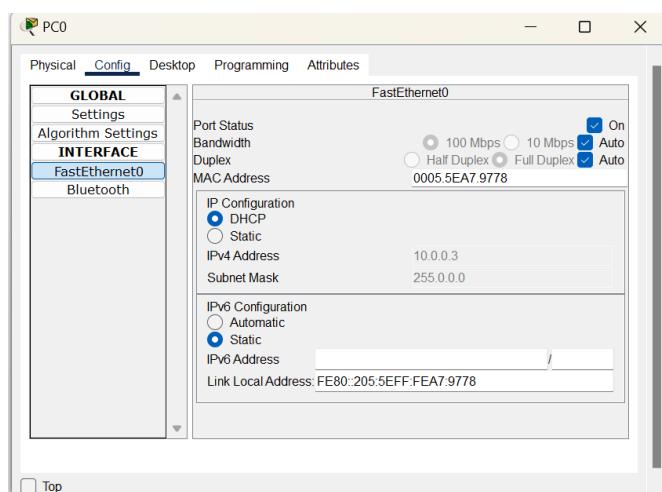


Figure 7.2: DHCP Service, PC0

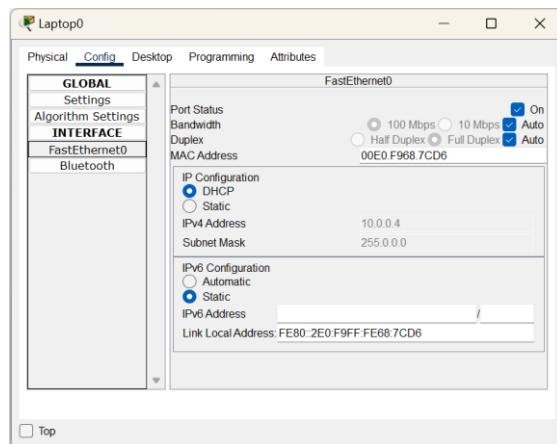


Figure 7.3: DHCP Service, Laptop0

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Laptop0	ICMP		0.000	N	0	(edit)	

PC0

Physical	Config	Desktop	Programming	Attributes
Command Prompt				
<pre>Cisco Packet Tracer PC Command Line 1.0 C:\&gt;ping 10.0.0.4  Pinging 10.0.0.4 with 32 bytes of data:  Reply from 10.0.0.4: bytes=32 time&lt;1ms TTL=128  Ping statistics for 10.0.0.4:     Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),     Approximate round trip times in milli-seconds:         Minimum = 0ms, Maximum = 0ms, Average = 0ms</pre>				

DATE: PAGE:

12-11-24

LABORATORY PROGRAM - 7(a)

Ques Aim Of The Experiment: To configure IP addresses of the host using DHCP server present within the same LAN.

Ans

TOPOLOGY :

CONNECTIONS : Copper Straight-Through

CONFIGURATION :

- Connect the topology using copper straight through
- Switch-PT Config:  
At fastethernet0 : IP address : 10.0.0.1  
Subnet mask : 255.0.0.0
- Gateway/DNS : Gateway : 10.0.0.2  
DNS Server : 10.0.0.1
- At Services : Use DHCP protocol :  
On the server.  
PoolName : Serverpool1

Ques (a) Default gateway : 10.0.0.2  
DNS Server : 10.0.0.1  
And have the DHCP  
At Router-PT : In FastEthernet0/0.  
IP address : 10.0.0.2  
Port Status : On.

Thus all the configuration are done.

Ans OBSERVATION :

Now, for the PCO and LAPTOP, the IP addresses are assigned automatically through DHCP protocol.

On clicking Gateway/DNS, the DHCP

- the PCO : fastethernet0 : IP Address : 10.0.0.4
- Laptop : fastethernet0 : IP Address : 10.0.0.3

Ques Ping from PCO > ping 10.0.0.4

Pinging 10.0.0.4 with 32 bytes of data:

Reply from 10.0.0.4 : bytes=32 time=1ms TTL=128

Reply from 10.0.0.4 : bytes=32 time=10ms TTL=128

Reply from 10.0.0.4 : bytes=32 time=1ms TTL=128

Reply from 10.0.0.4 : bytes=32 time=7ms TTL=128

Ping statistics for 10.0.0.4:

Packet: Sent=4, Received=4, Lost=0 (0% loss)  
Approx round trip times in milliseconds:  
Min=1ms, Max=10ms, Average=4ms

~~TX~~ 10.0.0.4 received q<  
~~RX~~ 10.0.0.4 received q<

## LABORATORY PROGRAM – 7(B)

To Configure IP addresses of the host using DHCP server outside a LAN.

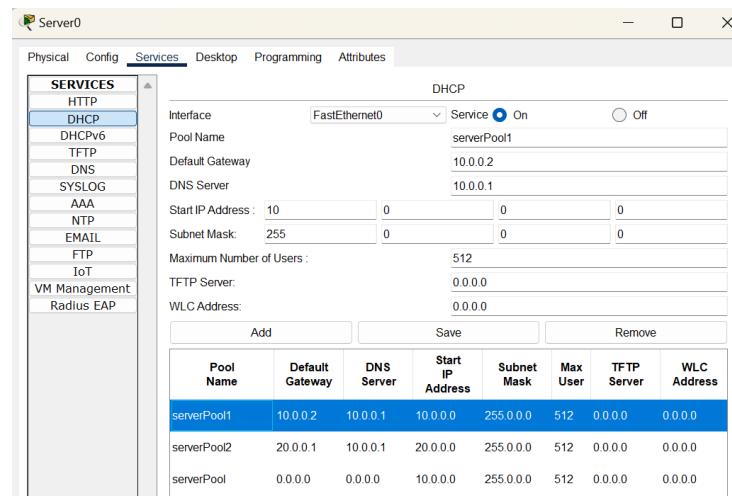
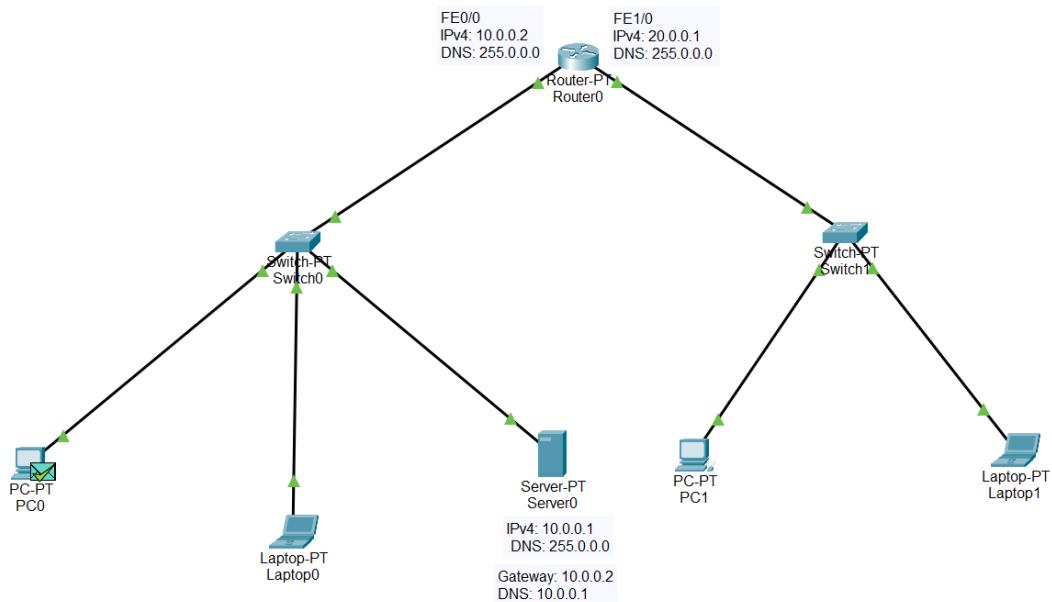


Figure 7.1.1: DHCP Service, Server0

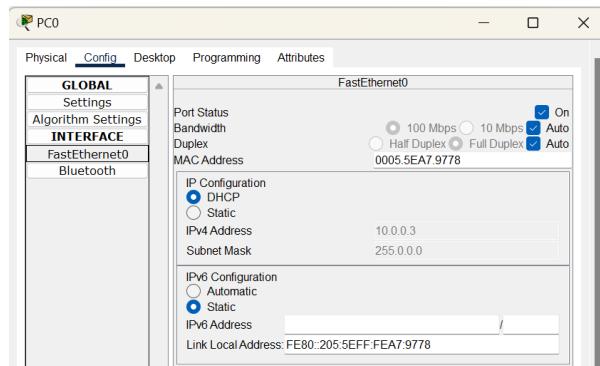


Figure 7.2.2: DHCP Service, PC0

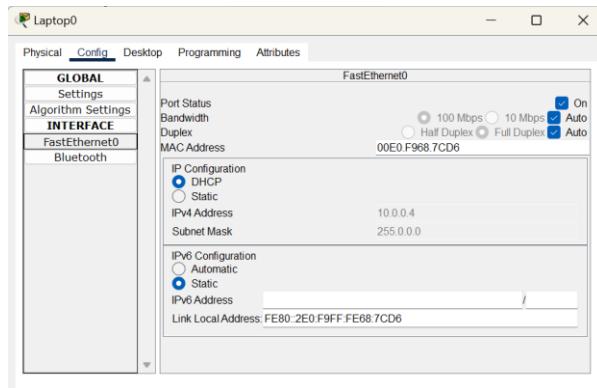


Figure 7.2.3: DHCP Service, Laptop0

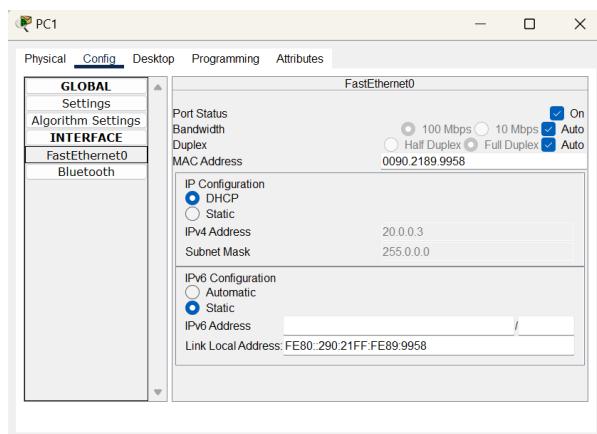


Figure 7.2.4: DHCP Service, PC1

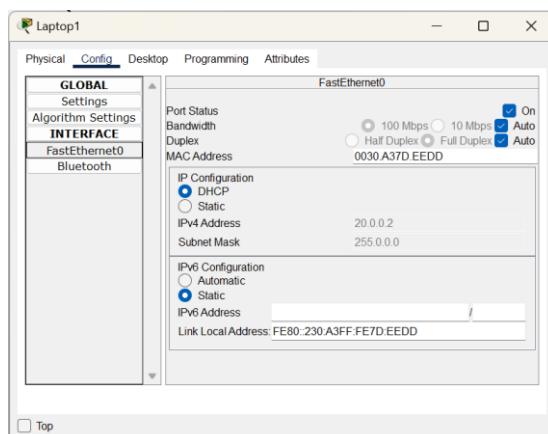


Figure 7.2.5: DHCP Service, Laptop1

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Laptop0	ICMP		0.000	N	0	(edit)	
	Successful	PC1	Laptop1	ICMP		0.004	N	1	(edit)	

LABORATORY PROGRAM - 7(B)

**AIM OF THE EXPERIMENT:** To config DHCP servers present outside the LAN.

**TOPOLOGY:**

**CONFIGURATION:**

- As per the topology, connections are made through copper cables straight through.
- And for the main network, the configuration is done as per previous experiment.
- And for the remote network, the config. Router: In fastEthernet1/0 IP address: 20.0.0.1 Port Status: On
- And > ip helper 20.0.0.1  
In fastEthernet0/0  
> ip helper 20.0.0.1

**Config Server 0:** At server1, using DHCP protocol: Service is On.

→ Pool Name: server1pool2  
Default Gateway: 20.0.0.1  
DNS Server: 10.0.0.1  
Start IP Address: 20.0.0.0  
Subnet Mask: 255.0.0.0  
Save the server1pool2.

→ Pool Name: server1pool1  
Default Gateway: 10.0.0.2  
DNS Server: 10.0.0.1  
Start IP Address: 10.0.0.0  
Save the server1pool1.

Thus all the configuration are done.

**OBSERVATIONS:**

- Now, on clicking DHCP in PC1:  
In fastEthernet0/0: IP address: 20.0.0.2 Subnet Mask: 255.0.0.0
- On clicking DHCP in Laptop 1:  
In fastEthernet0/0: IP address: 20.0.0.3 Subnet Mask: 255.0.0.0
- And similarly for the PC2, Laptop2, the IP address gets configured automatically.
- In PC2> ping 20.0.0.2  
Ping to 20.0.0.2 with 32 bytes of data:  
Reply from 20.0.0.2: bytes=32 time<1ms TTL=128

Ping statistics for 20.0.0.2:  
bytes: Sent=6 Received=4 Lost=0 (0% loss).  
Min=0ms, Max=0ms, Average=0ms

## LABORATORY PROGRAM – 8

To Configure DNS server to demonstrate the mapping of IP addresses and Domain names.

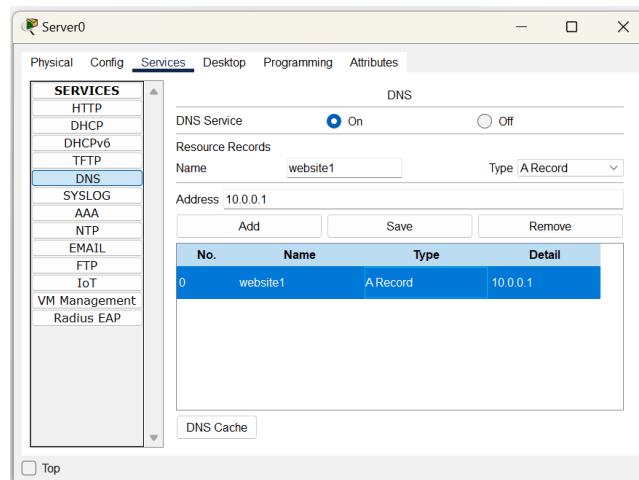
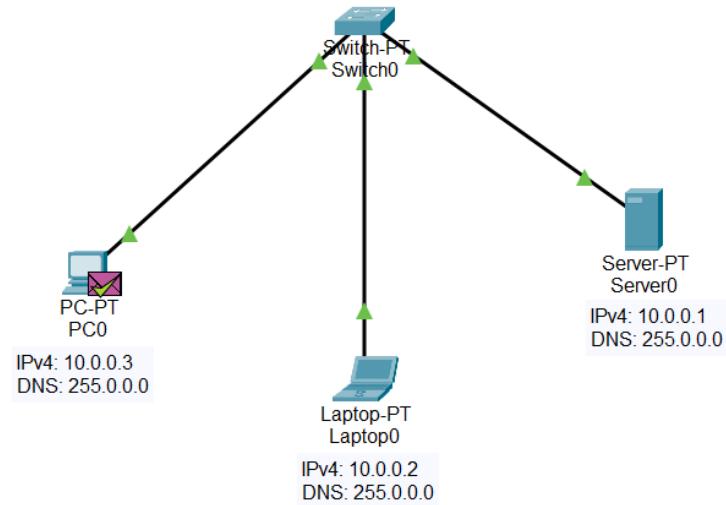


Figure 8.1: DNS Service, Server0

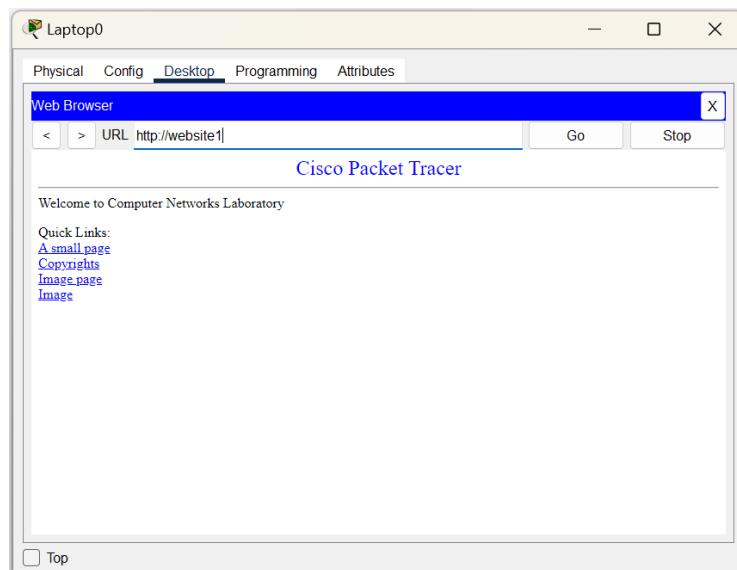


Figure 8.2: DNS Service, Laptop0

DATE	PAGE
18-11-24	
	Laboratory Program - 8 (Ans)
Ans	Aim Of The Experiment :- To configure DNS server to demonstrate the mapping of IP addresses and domain names.
Ans	POPOLGY :
Ans	CONFIGURATION :
o	Connection : Copper straight through
o	At SourceID : 1234
o	In fastEthernet0 : IP address : 10.0.0.1
o	In services > Use DHCP protocol.
o	Default Gateway : 0.0.0.0
o	DNS Server : 10.0.0.10
>	In DNS Services : Name : webserver
>	Address : 10.0.0.1
>	In HTTP :
o	Edit index.html, as per your requirement.
	URL : http://webserver:1
	Cisco Packet Tracer
	Welcome to Computer Networks Laboratory
	Quick links:
	A small page
	Copyright
	Image page (in pleasant resolution)
	Image
	Index
	Logout
	Help

## LABORATORY PROGRAM – 9

To Configure RIP routing protocol in Routers.

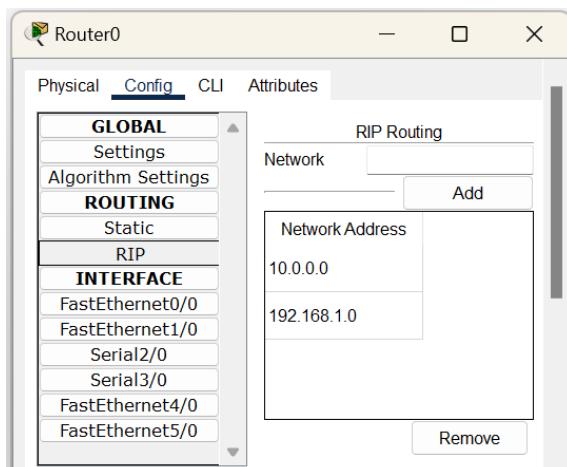
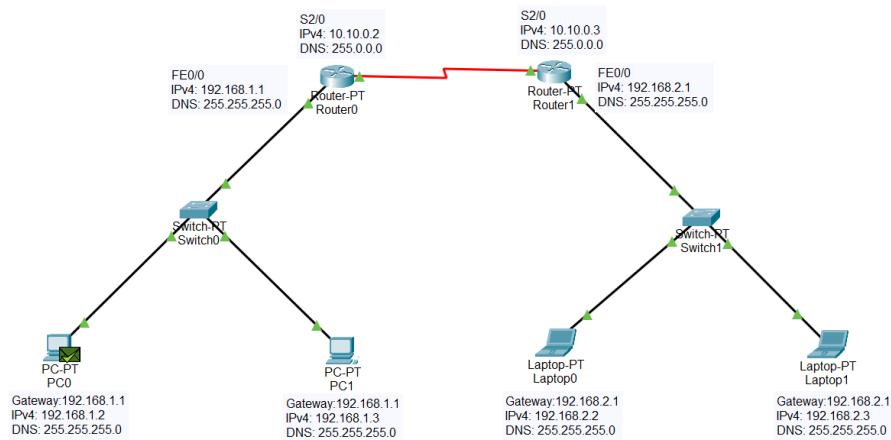


Figure 9.1: RIP, Router0

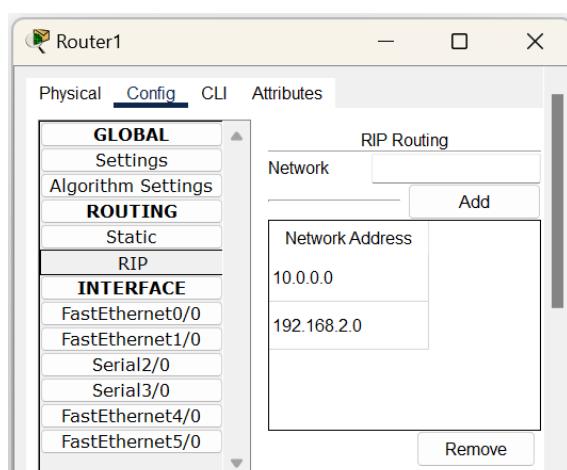


Figure 9.2: RIP, Router1

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Laptop1	ICMP		0.000	N	0	(edit)	

```

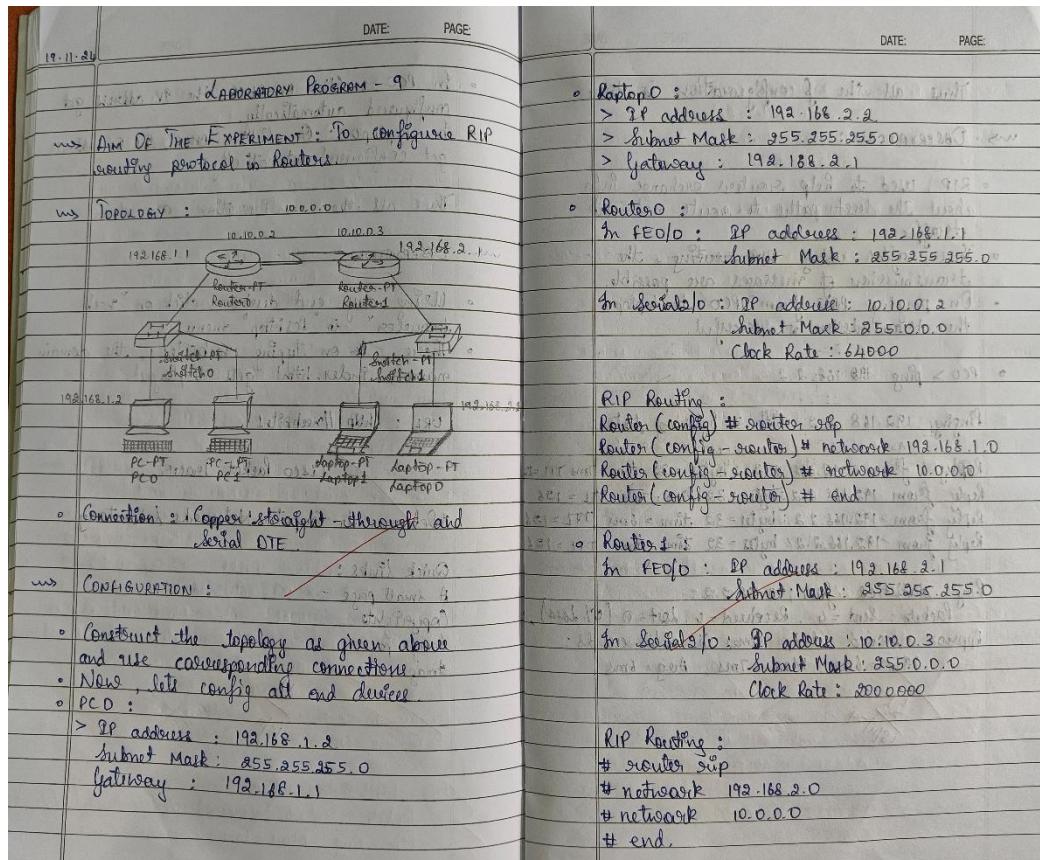
C:\>ping 192.168.2.3

Pinging 192.168.2.3 with 32 bytes of data:

Reply from 192.168.2.3: bytes=32 time=18ms TTL=126
Reply from 192.168.2.3: bytes=32 time=14ms TTL=126
Reply from 192.168.2.3: bytes=32 time=1ms TTL=126
Reply from 192.168.2.3: bytes=32 time=1ms TTL=126

Ping statistics for 192.168.2.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 18ms, Average = 8ms

```



Thus all the configuration is done.

OUR OBSERVATION :

- RIP, used to help routers exchange info about the best path to route network traffic.
- Thus, after using RIP routing, the transmission of messages are possible.
- On pinging PC0 from PC0 to Laptop A, the transmission was successful.
- PC0 > ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data::

Reply from 192.168.2.2 : bytes = 32 time = 7ms TTL = 126  
 Reply from 192.168.2.2 : bytes = 32 time = 7ms TTL = 126  
 Reply from 192.168.2.2 : bytes = 32 time = 6ms TTL = 126  
 Reply from 192.168.2.2 : bytes = 32 time = 6ms TTL = 126

Ping statistics for 192.168.2.2:

Packet: Sent = 4, Received = 4, Lost = 0 (0% loss),  
 Approx round trip times in milli-seconds:

Min = 6ms, Max = 7ms, Avg = 6ms

~~BT  
18/11/19~~

## LABORATORY PROGRAM – 10

To demonstrate communication between two devices using a wireless LAN.

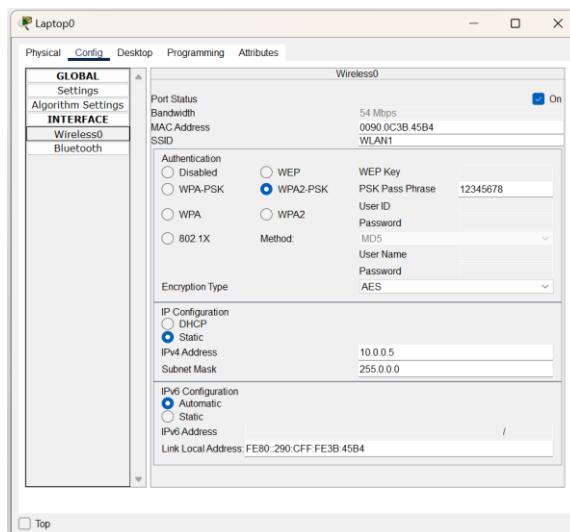
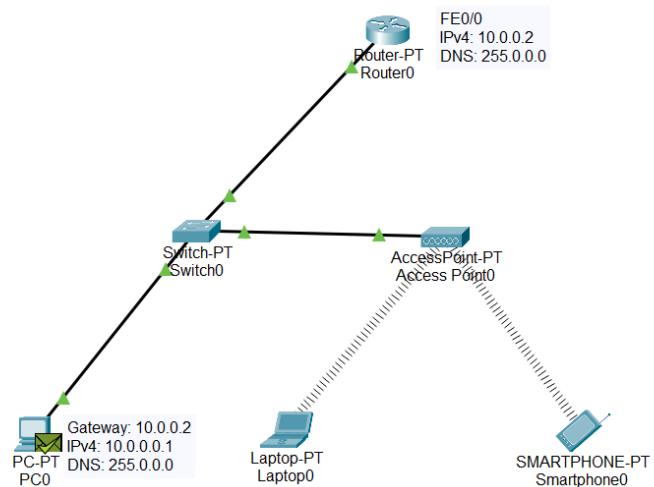


Figure 10.1: Laptop0, Wireless0

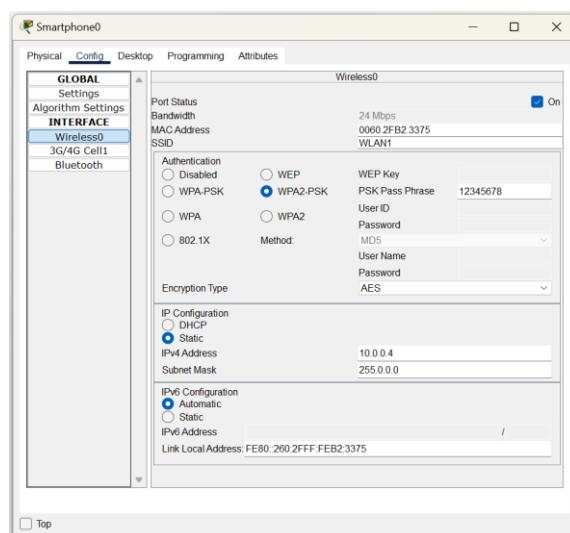


Figure 10.2: Smartphone0, Wireless0

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Laptop0	ICMP	Green	0.000	N	0	(edit)	

PC0

Physical Config Desktop Programming Attributes

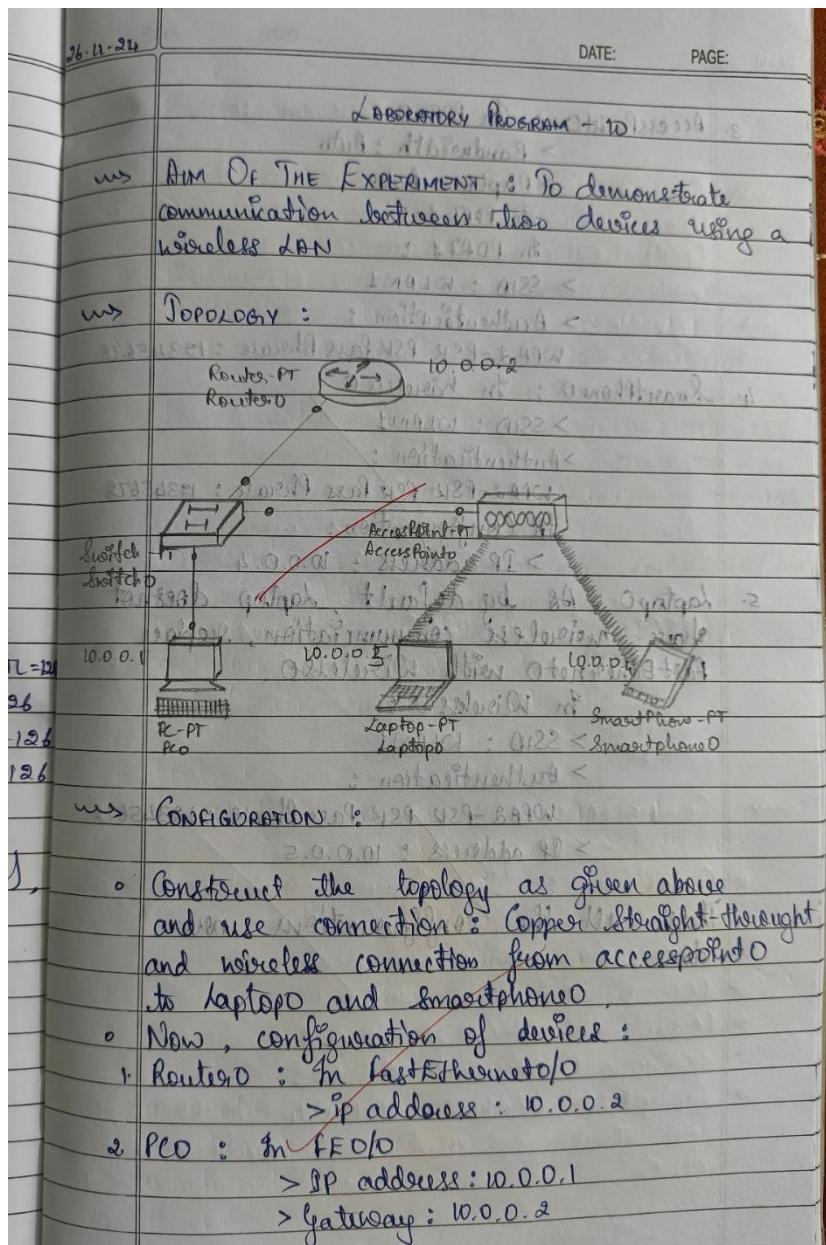
Command Prompt

```
Cisco Packet Tracer PC Command Line 1.0
C:>ping 10.0.0.5

Pinging 10.0.0.5 with 32 bytes of data:

Reply from 10.0.0.5: bytes=32 time=8ms TTL=128
Reply from 10.0.0.5: bytes=32 time=28ms TTL=128
Reply from 10.0.0.5: bytes=32 time=30ms TTL=128
Reply from 10.0.0.5: bytes=32 time=36ms TTL=128

Ping statistics for 10.0.0.5:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 8ms, Maximum = 36ms, Average = 25ms
```



	DATE: _____ PAGE: _____
<p>3. AccessPoint0 : In PORT 0 :</p> <ul style="list-style-type: none"> <li>&gt; Bandwidth : Auto</li> <li>&gt; Duplex : Auto</li> <li>&gt; In PORT 3 :</li> <li>&gt; In PORT 4 :</li> <li>&gt; SSID : WLAN1</li> <li>&gt; Authentication :</li> <li>WPA2-PSK PSK Pass Phrase : 12345678</li> </ul> <p>4. Smartphone0 : In Wireless0 :</p> <ul style="list-style-type: none"> <li>&gt; SSID : WLAN1</li> <li>&gt; Authentication :</li> <li>WPA2-PSK PSK Pass Phrase : 12345678</li> <li>IP Configuration :</li> <li>&gt; IP address : 10.0.0.4</li> </ul> <p>5. Laptop0 : As by default, laptop does not have wireless communication, replace fast Ethernet0 with Wireless0.</p> <p>In Wireless0 :</p> <ul style="list-style-type: none"> <li>&gt; SSID : WLAN1</li> <li>&gt; Authentication :</li> <li>WPA2-PSK PSK Pass Phrase : 12345678</li> <li>&gt; IP address : 10.0.0.5</li> </ul> <p>Thus, all the configurations are done. Now, let's check if the connection has been established or not. Open terminal and type:</p> <pre>ping 10.0.0.4</pre> <pre>Min = 17ms, Max = 31ms, Avg = 23ms</pre>	<p>DATE: _____ PAGE: _____</p> <p>→ OBSERVATION :</p> <ul style="list-style-type: none"> <li>• As we have successfully established wireless LAN (communication), now if it is possible to ping messages from any device.</li> <li>• Pinging message from Laptop0 to Smartphone0</li> <li>&gt; Ping 10.0.0.4</li> <li>Pinging 10.0.0.4 with 32 bytes of data :</li> <li>Reply from 10.0.0.4 : bytes=32 time=31ms TTL=128</li> <li>Reply from 10.0.0.4 : bytes=32 time=17ms TTL=128</li> <li>Reply from 10.0.0.4 : bytes=32 time=23ms TTL=128</li> <li>Reply from 10.0.0.4 : bytes=32 time=18ms TTL=128</li> <li>Time statistics from 10.0.0.4 :</li> <li>Packets : Sent = 4, Received = 4, Lost = 0 (0% loss)</li> <li>Average round trip times in ms :</li> <li>Min = 17ms, Max = 31ms, Avg = 23ms</li> </ul> <p><del>RTT</del> <del>RTT</del></p>

## LABORATORY PROGRAM – 11

To demonstrate the working of Address Resolution Protocol (ARP) within a LAN for communication.

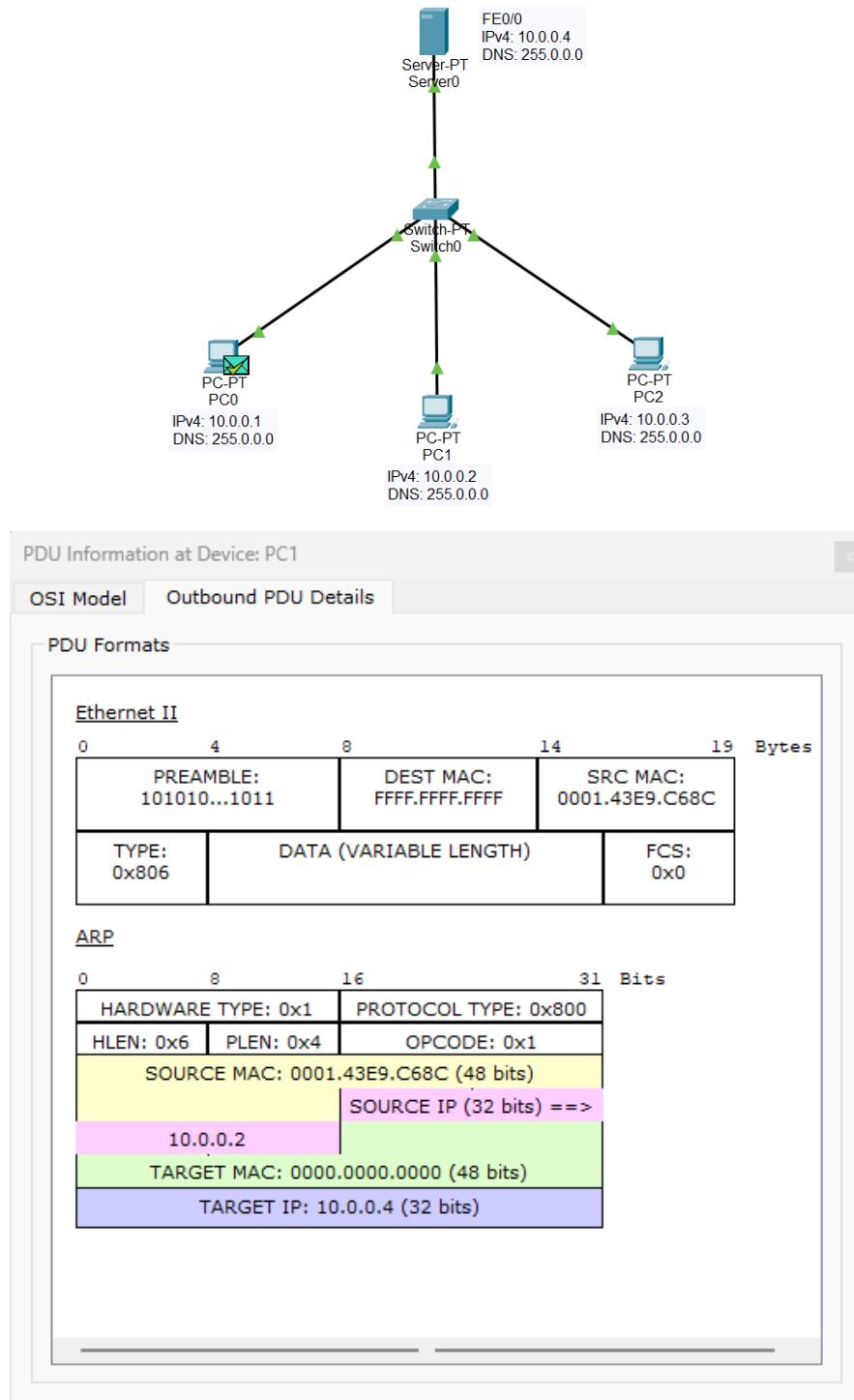


Figure 11.1: Inbound ARP, PC1

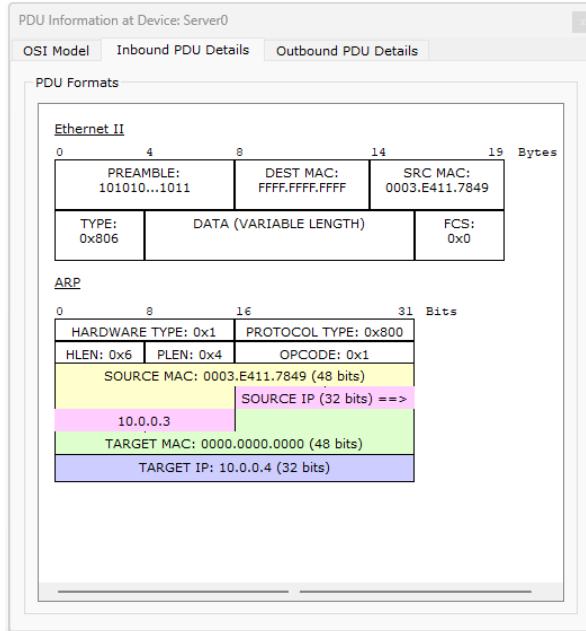


Figure 11.2: Inbound ARP, Server0

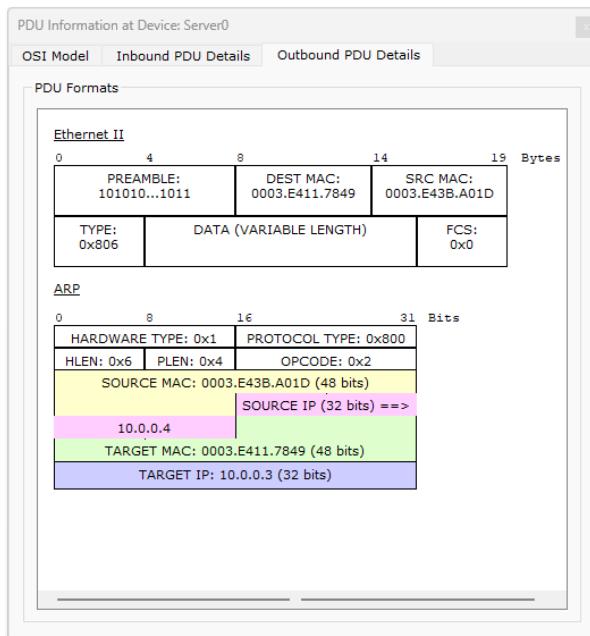


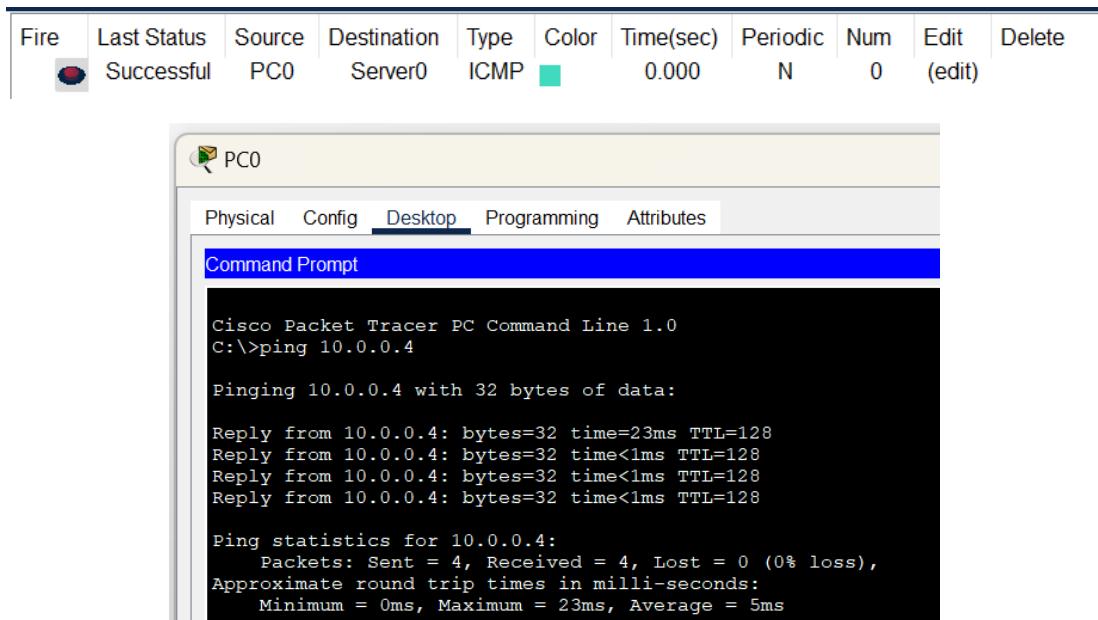
Figure 11.3: Outbound ARP, Server0

ARP Table for Server0			
IP Address	Hardware Address	Interface	
10.0.0.1	00E0.B062.0C32	FastEthernet0	
10.0.0.2	0001.43E9.C68C	FastEthernet0	

Figure 11.4: ARP Table, Server0

IP Address	Hardware Address	Interface
10.0.0.4	0003.E43B.A01D	FastEthernet0

Figure 11.5: ARP Table, PC1



LABORATORY PROGRAM - 11/11/2021

**QUESTION:**

Ques. Aim Of This Experiment : To demonstrate the working of address resolution protocol (ARP) within a LAN for communication.

**ANSWER:**

→ **Topology :**

```

graph TD
    Router[Router-PC1] --- S1[Switch]
    S1 --- PC1[PC1-PC2]
    S1 --- PC2[PC2-PC3]
    S1 --- Router[Router-PC3]
    Router --> PC1
    Router --> PC2
    Router --> PC3
    
```

→ **Configuration :**

- Connection : Use Copper straight-through for connection.
- Now, Configuring end devices :
  - > PC0 : IP address : 10.0.0.1
  - > PC1 : IP address : 10.0.0.2
  - > PC2 : IP address : 10.0.0.3
  - > Router : IP address : 10.0.0.4 (In RCD/D)
 Thus connections and configurations are done.

**OBSERVATION :**

- Initially, if we prompt : > ARP -a, the ARP table is empty.
- Now lets PDU from PC1 to destination, on clicking PDU, type ARP : we get ARP table describing :
  - SOURCE MAC : 0000.0000.0000 (48 bits)
  - SOURCE IP : 10.0.0.2
  - TARGET MAC : 0000.0000.0000 (48 bits)
  - TARGET IP : 10.0.0.4 (32 bits)
- Now, on clicking PDU, type ARP of the Router : we get ARP table describing :
  - SOURCE MAC : 0003.E811.7869 (48 bits)
  - SOURCE IP : 10.0.0.4
  - TARGET MAC : 0003.E811.7869 (48 bits)
  - TARGET IP : 10.0.0.3 (32 bits)

Thus ARP table for each device gets filled. Now communication is possible.

## LABORATORY PROGRAM – 12

To create a VLAN on top of the physical LAN and enable communication between physical LAN and virtual LAN.

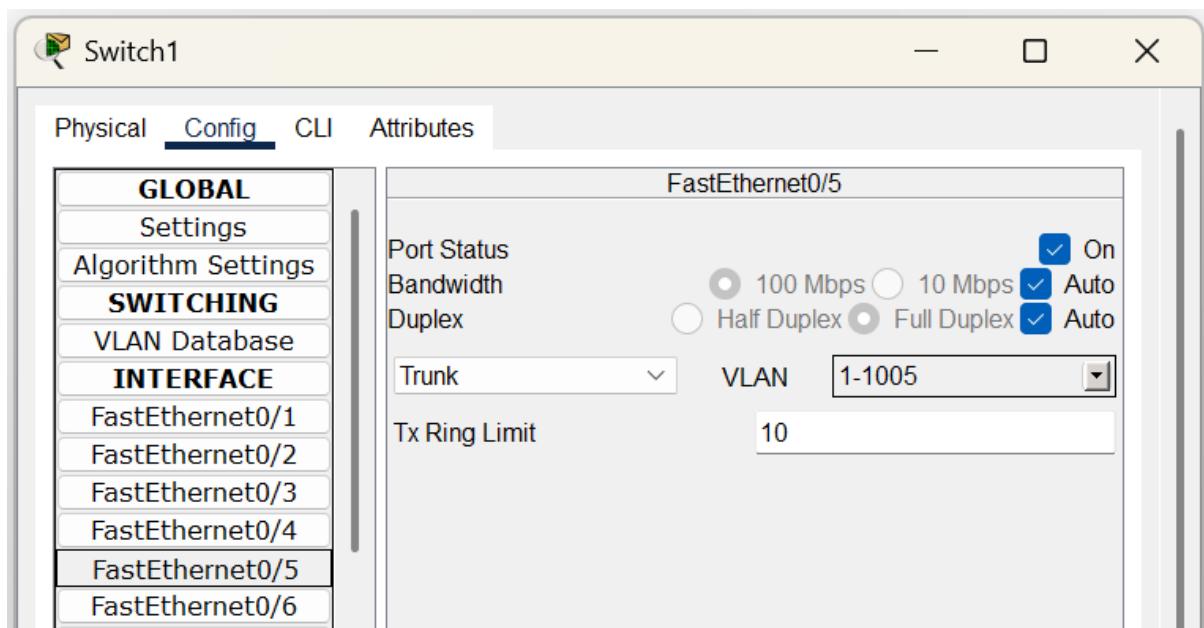
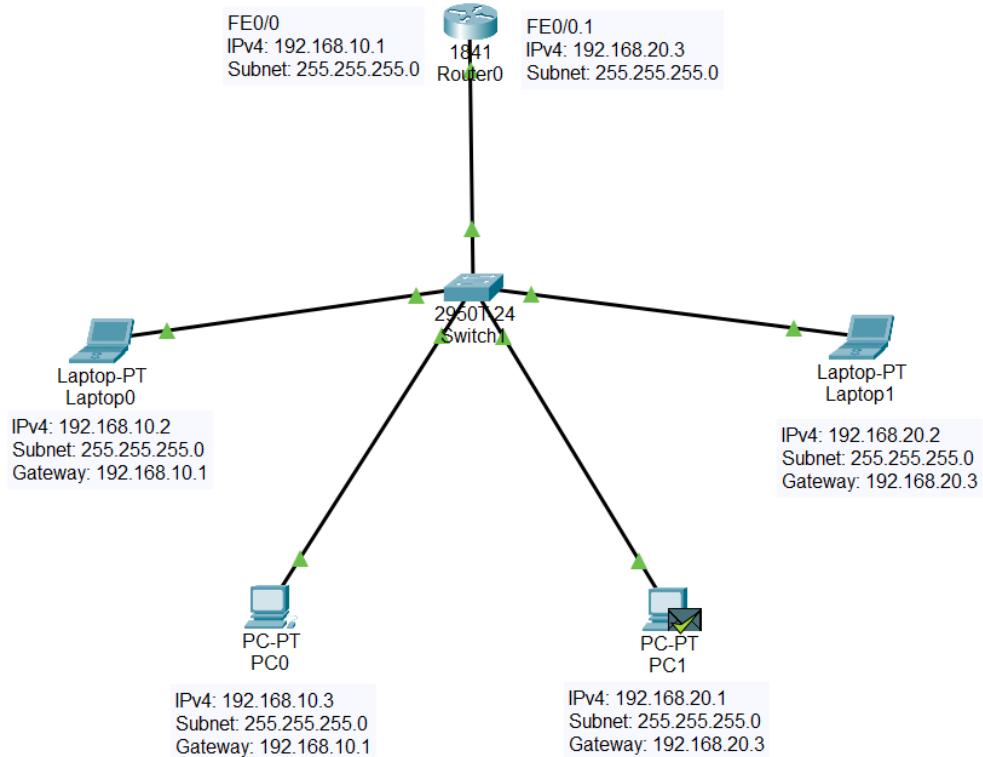


Figure 12.1: FE0/5 Switchport Trunk

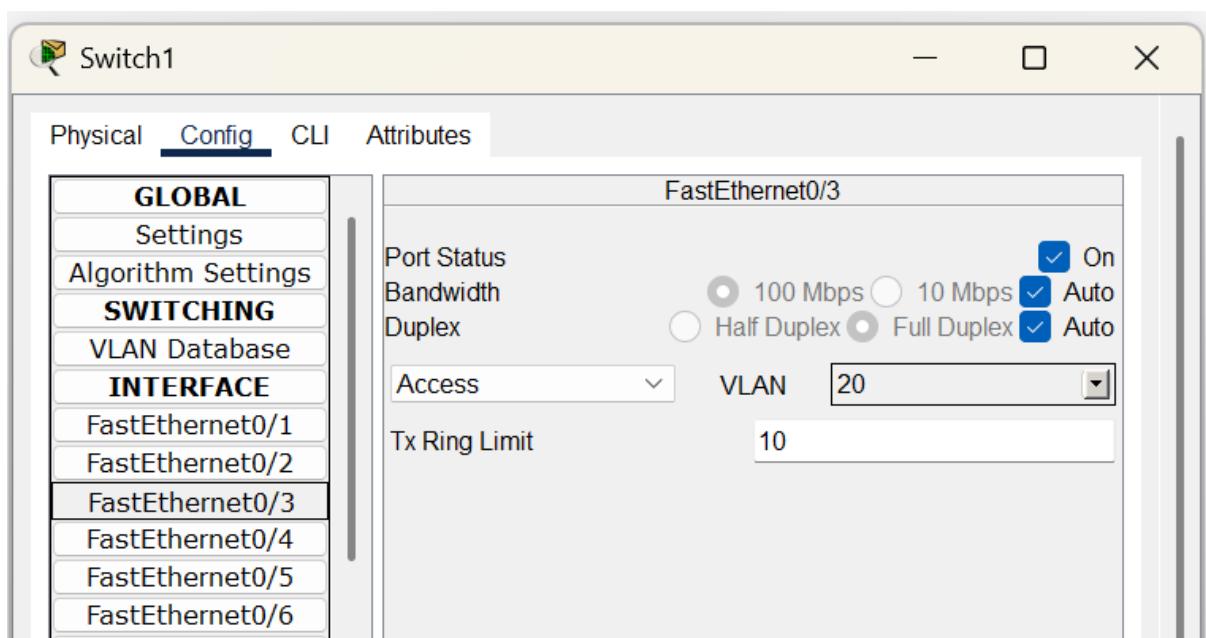


Figure 12.2: FEO/3 Switchport Access

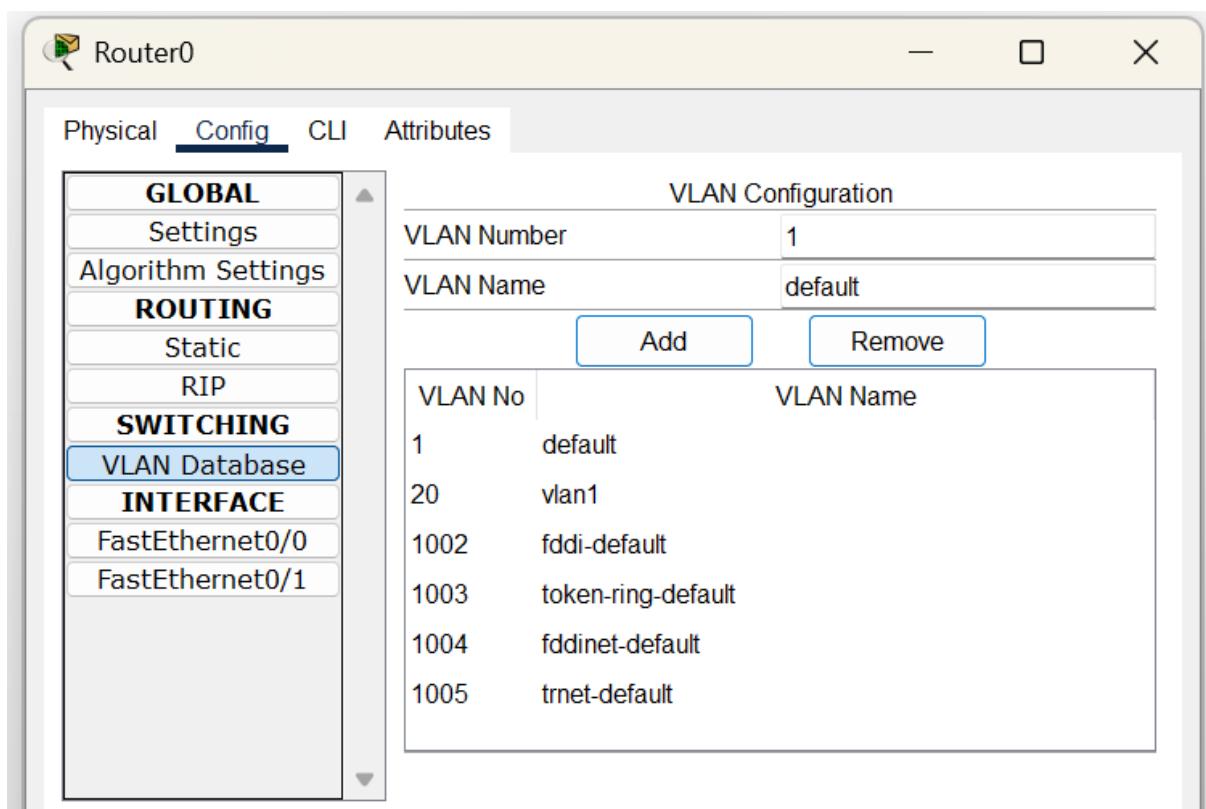


Figure 12.3: Router0 VLAN Database

```

Router(config)#interface FastEthernet0/0.1
Router(config-subif)#encapsulation dot1q 20
Router(config-subif)#ip address 192.168.20.3 255.255.255.0
Router(config-subif)#no shutdown

```

Figure 2: Router0, FEO/0.1

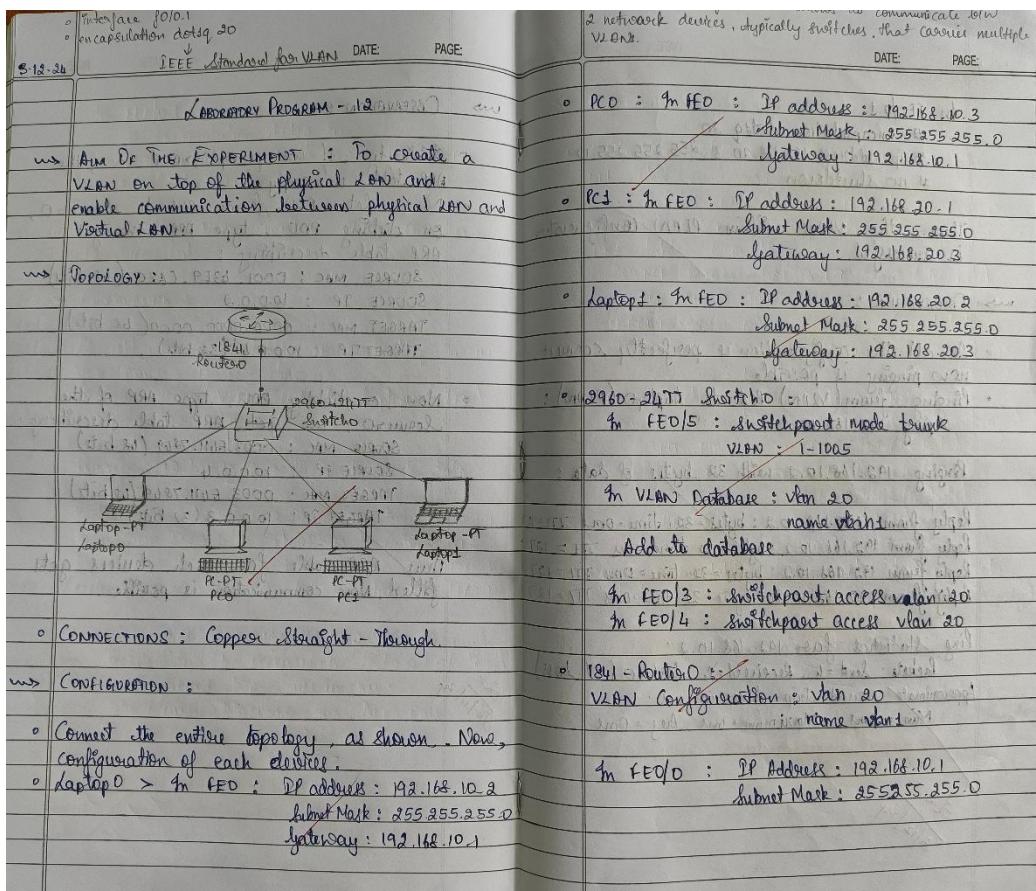
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC1	Router0	ICMP		0.000	N	0	(edit)	

```
C:\>ping 192.168.20.3

Pinging 192.168.20.3 with 32 bytes of data:

Reply from 192.168.20.3: bytes=32 time=2ms TTL=255
Reply from 192.168.20.3: bytes=32 time<1ms TTL=255
Reply from 192.168.20.3: bytes=32 time<1ms TTL=255
Reply from 192.168.20.3: bytes=32 time<1ms TTL=255

Ping statistics for 192.168.20.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 2ms, Average = 0ms
```



In FEO/0.1 : member 9C : 077 ms : 0.99  
 # encapsulation dot1q 20  
 # ip address 192.168.20.3 255.255.255.0  
 # no shutdown

Thus, the VLAN within PLAN configuration is successful.

~~now Observation : member 9C : 077 ms : 0.99~~

- As all the configuration is perfectly correct, now ping is possible.
- Pinging from (VLAN) PC1 to Laptop (PLan) :
   
ping 192.168.10.2 : 0.037 ms

Pinging 192.168.10.2 with 32 bytes of data :

Reply from 192.168.10.2 : bytes = 32 time = 0ms TTL = 127

Reply from 192.168.10.2 : bytes = 32 time = 1ms TTL = 127

Reply from 192.168.10.2 : bytes = 32 time = 0ms TTL = 127

Reply from 192.168.10.2 : bytes = 32 time = 0ms TTL = 127

Ping statistics for 192.168.10.2 :

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)

Approximate round trip times in ms :

Min = 0ms, Max = 1ms, Avg = 0ms

192.168.10.2 : member 9C : 0.037 ms

~~RTT = 0.037 ms~~

~~RTT = 0.037 ms~~

## LABORATORY PROGRAM – 13

Write a program for error detecting code using CRC-CCITT (8-bits).

### Code

```
def xor(dividend, divisor):
    """Perform XOR operation between dividend and divisor."""
    result = ""
    for i in range(1, len(divisor)):
        result += '0' if dividend[i] == divisor[i] else '1'
    return result

def crc(data, gen_poly):
    """Compute the CRC check value using CRC-CCITT (8-bit)."""
    data_length = len(data)
    gen_length = len(gen_poly)

    # Append n-1 zeros to the data
    padded_data = data + '0' * (gen_length - 1)
    check_value = padded_data[:gen_length]

    for i in range(data_length):
        if check_value[0] == '1':
            # XOR operation if the first bit is 1
            check_value = xor(check_value, gen_poly)
        else:
            # Retain original check value if first bit is 0
            check_value = check_value[1:]

        # Shift left and add the next data bit
        if i + gen_length < len(padded_data):
            check_value += padded_data[i + gen_length]

    return check_value[1:] # Remove the leading bit

def receiver(data, gen_poly):
    """Simulate the receiver side to check for errors."""
    print("\n-----")
    print("Data received:", data)

    # Perform CRC computation on received data
    remainder = crc(data, gen_poly)

    # Check if the remainder is all zeros
    if '1' in remainder:
        print("Error detected")
    else:
        print("No error detected")

if __name__ == "__main__":
```

```

# Input data and generator polynomial
data = input("Enter data to be transmitted: ")
gen_poly = input("Enter the Generating polynomial: ")

# Compute CRC check value
check_value = crc(data, gen_poly)
print("\n-----")
print("Data padded with n-1 zeros:", data + '0' * (len(gen_poly) - 1))
print("CRC or Check value is:", check_value)

# Append check value to data for transmission
transmitted_data = data + check_value
print("Final data to be sent:", transmitted_data)
print("-----\n")

# Simulate the receiver side
received_data = input("Enter the received data: ")
receiver(received_data, gen_poly)

```

### Output

```

Enter data to be transmitted: 1001100
Enter the Generating polynomial: 100001011

-----
Data padded with n-1 zeros: 1001100000000000
CRC or Check value is: 0100010
Final data to be sent: 10011000100010

-----
Enter the received data: 10011000100011

-----
Data received: 10011000100011
Error detected

```

## LABORATORY PROGRAM - 13

Write a program for error detecting code using CRC-CCITT (8-bits)

```
def xor(dividend, divisor):
    result = ''
    for i in range(1, len(divisor)):
        result += '0' if dividend[i] == divisor[i] else '1'
    return result
```

```
def crc(data, gen_poly):
    data_length = len(data)
    gen_length = len(gen_poly)
```

```
padded_data = data + '0' * (gen_length - 1)
check_value = padded_data[:gen_length]
for i in range(data_length+1):
    if check_value[0] == '1':
        check_value = xor(check_value, gen_poly)
    else:
        check_value = check_value[1:]
```

```
if i + gen_length < len(padded_data):
    check_value += padded_data[i+gen_length]
return check_value[1:]
```

```

def receiver(data, gen_poly):
    print("\n-----")
    print("Data received: ", data)
    remainder = crc(data, gen_poly)
    if '1' in remainder:
        print("Error detected")
    else:
        print("No error detected")

if name == "main":
    data = input("Enter data to be transmitted: ")
    gen_poly = input("Enter the generating polynomial: ")
    check_value = crc(data, gen_poly)
    print("Data padded with n-1 zeros: ", data + '0' * (len(gen_poly) - 1))
    print("CRC or Check value is: ", check_value)

transmitted_data = data + check_value
print("Final data to be sent: ", transmitted_data)
print("\n")

```

Received data: Enter the received data: 10011000100011

DATE: PAGE: 31.01.19 PAGE: 31.01.19

Enter data to be transmitted: 10011000100011  
Enter the generating polynomial: 100001011

Data padded with n-1 zeros: 1001100000000000  
CRC or Check value is: 01000100010

Final data to be sent: 100110001000100010

Enter the received data: 10011000100011

Data received: 10011000100011

Error detected

How does the CRC obtained? The data to sent is adjusted with n-1 zeroes where n is the length of key then binary division of the adjunct data by key is done the remainder is the checksum code, that is XORed with adjunct data.

~~Explain how to work by diagram~~

~~What is the role of CRC?~~

~~What is the difference between CRC and checksum?~~

~~What is the role of parity bit?~~

~~What is the role of FCS?~~

~~What is the role of FCS?~~

## LABORATORY PROGRAM – 14

Write a program for congestion control using Leaky bucket algorithm.

### Code

```
# Getting user inputs
storage = int(input("Enter initial packets in the bucket: "))
no_of_queries = int(input("Enter total no. of times bucket content is checked: "))
bucket_size = int(input("Enter total no. of packets that can be accommodated in the bucket: "))
input_pkt_size = int(input("Enter no. of packets that enters the bucket at a time: "))
output_pkt_size = int(input("Enter no. of packets that exits the bucket at a time: "))

for i in range(no_of_queries): # space left
    size_left = bucket_size - storage
    if input_pkt_size <= size_left:
        # update storage
        storage += input_pkt_size
    else:
        print("Packet loss =", input_pkt_size)

print(f"Buffer size = {storage} out of bucket size = {bucket_size}")

# as packets are sent out into the network, the size of the storage decreases
storage -= output_pkt_size
```

### Output

```
Enter initial packets in the bucket: 0
Enter total no. of times bucket content is checked: 4
Enter total no. of packets that can be accommodated in the bucket: 10
Enter no. of packets that enters the bucket at a time: 4
Enter no. of packets that exits the bucket at a time: 1
Buffer size = 4 out of bucket size = 10
Buffer size = 7 out of bucket size = 10
Buffer size = 10 out of bucket size = 10
Packet loss = 4
Buffer size = 9 out of bucket size = 10
```

<p>DATE: 17-12-26 PAGE: 32</p> <p>LABORATORY PROGRAM - 14</p> <p>Write a program for congestion control, using leaky bucket algorithm.</p> <pre> storage = int(input("Enter initial packets in the bucket :")) no_of_packets = int(input("Enter total no. of times bucket content is checked :")) bucket_size = int(input("Enter total no. of packets that can be accommodated in the bucket :")) input_pkt_size = int(input("Enter no. of packets that enter the bucket at a time :")) output_pkt_size = int(input("Enter no. of packets that exits the bucket at a time :"))  for i in range (no_of_packets):     size_left = bucket_size - storage     if input_pkt_size &lt;= size_left:         storage += input_pkt_size     else:         print("Packet loss = ", input_pkt_size)     print(f"Buffer size = {storage} out of bucket size = {bucket_size}")  storage = output_pkt_size </pre>	<p>DATE: PAGE: 33</p> <p>us OUTPUT:</p> <p>Enter initial packets in the bucket : 0</p> <p>Enter total no. of times bucket content is checked : 4</p> <p>Enter total no. of packets that can be accommodated in the bucket : 10</p> <p>Enter no. of packets that enter the bucket at a time : 4</p> <p>Enter no. of packets that exits the bucket at a time : 1</p> <p>Buffer size = 4 out of bucket size = 10</p> <p>Buffer size = 7 out of bucket size = 10</p> <p>Buffer size = 10 out of bucket size = 10</p> <p>Bucket loss = 4</p> <p>Buffer size = 9 out of bucket size = 10</p> <p>(input pkt size) &gt;= (bucket size)</p> <p>(input pkt size) &lt; (bucket size) = (packet loss)</p> <p>(input pkt size) &lt; (bucket size) = (packet loss)</p> <p>* temporary buffer</p> <p>1. Loss = 0.000000 2. Loss = 0.250000 3. Loss = 0.500000 4. Loss = 0.750000</p>
---	---

## LABORATORY PROGRAM – 15(A)

Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

### Code: Client.py

```
from socket import *
serverName = "127.0.0.1" # Server address (localhost)
serverPort = 12000 # Port number where the server listens

# Create TCP socket
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort)) # Connect to server

# Ask user for file name to request
sentence = input("Enter file name: ")

# Send file name to server
clientSocket.send(sentence.encode())

# Receive file contents from server
filecontents = clientSocket.recv(1024).decode()
print('From Server:', filecontents)

# Close the connection
clientSocket.close()
```

### Code: Server.py

```
from socket import *
serverName = "127.0.0.1" # Server address (localhost)
serverPort = 12000 # Port number to listen on

# Create TCP socket
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort)) # Bind socket to the address and port
serverSocket.listen(1) # Listen for 1 connection
print("The server is ready to receive")

while True:
    # Accept a connection
    connectionSocket, addr = serverSocket.accept()

    # Receive the file name from the client
    sentence = connectionSocket.recv(1024).decode()

    # Try opening the file
    try:
        file = open(sentence, "r") # Open file in read mode
        fileContents = file.read(1024) # Read file content (up to 1024 bytes)
```

```

connectionSocket.send(fileContents.encode()) # Send file contents to client
file.close()
except FileNotFoundError:
    # Send error message if file not found
    connectionSocket.send("File not found".encode())

# Close the connection
connectionSocket.close()

```

## Output

```

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS SEARCH ERROR COMMENTS
(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)> py Client.py
Enter file name: TCP.txt
From Server: This is a test file.

Using TCP/IP sockets, write a client-server program to make client sending the
file name and the server to send back the contents of the requested file if
present.

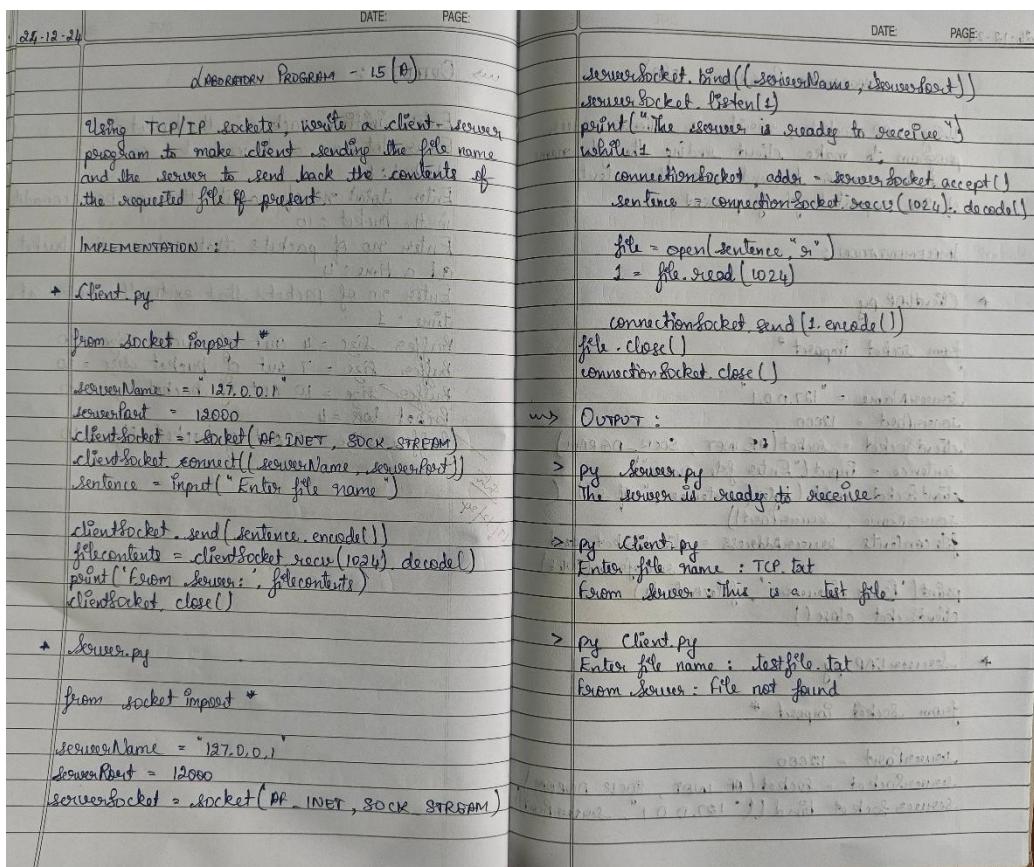
(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)> py Client.py
Enter file name: testfile.txt
From Server: File not found
(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)>

```

```

(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)> py Server.py
The server is ready to receive

```



## **LABORATORY PROGRAM – 15(B)**

Using UDP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

### **Code: ClientUDP.py**

```
from socket import *
serverName = "127.0.0.1" # Server address (localhost)
serverPort = 12000 # Port number where the server listens

# Create UDP socket
clientSocket = socket(AF_INET, SOCK_DGRAM)

# Ask user for file name to request
sentence = input("Enter file name: ")

# Send the file name to the server using UDP
clientSocket.sendto(sentence.encode("utf-8"), (serverName, serverPort))

# Receive file contents from the server
fileContents, serverAddress = clientSocket.recvfrom(2048)

# Print the file contents received from the server
print("From Server:", fileContents.decode())

# Close the UDP socket
clientSocket.close()
```

### **Code: ServerUDP.py**

```
from socket import *
serverPort = 12000 # Port number to listen on

# Create UDP socket
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort)) # Bind the socket to the server address and port

print("The server is ready to receive")

while True:
    # Receive file name from the client
    sentence, clientAddress = serverSocket.recvfrom(2048)

    # Try opening the file
    try:
        file = open(sentence.decode(), "r") # Open file in read mode
        fileContents = file.read(2048) # Read file content (up to 2048 bytes)
        serverSocket.sendto(fileContents.encode("utf-8"), clientAddress) # Send file contents to client
        file.close()
    except FileNotFoundError:
        pass
```

```
# Send error message if file not found
serverSocket.sendto("File not found".encode("utf-8"), clientAddress)
```

## Output

```
(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)> py ClientUDP.py
Enter file name: UDP.txt
From Server: This is a test file.

Using UDP sockets, write a client-server program to make client sending
the file
name and the server to send back the contents of the requested file if p
resent.

(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)> py ClientUDP.py
Enter file name: testfile.txt
From Server: File not found

(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)>
```

```
(base) PS D:\BMSCE\Fifth SEM CSE\CN\Lab - 15(24.12.24)> py ServerUDP.py
The server is ready to receive
```

